MASTER EN SCIENCES INFORMATIQUES

Apprentissage automatique. Application en Ingénierie des protéines

MONNEAUX, Jean-Claude; Lambert, Christophe

Award date:
2009

Académie « Louvain »

Facultés Universitaires Notre-Dame de la Paix
Institut d'informatique

Apprentissage automatique
Application
en Ingénierie des protéines

Jean-Claude Monneaux

Travail de fin d'études présenté pour l'obtention du grade de
Licencié en Informatique

JUIN 2009

RUE GRANDGAGNAGE, 21 • B-5000 NAMUR (BELGIUM)

APPRENTISSAGE AUTOMATIQUE
Application en ingénierie des protéines

Résumé

Les algorithmes d'apprentissage automatique (*Machine Learning*) sont principalement d'application dans des domaines où d'importantes quantités d'informations sont disponibles. Ces algorithmes sont exploités pour classifier ces données, pour estimer une fonction, pour découvrir des relations, pour simuler des lois des domaines étudiés.

On posera les bases théoriques des principaux systèmes d'intelligence artificielle et de leurs algorithmes d'apprentissage. Les limites d'applicabilité et la validité des résultats induits seront examinées.

Depuis le séquençage du génome humain, la quantité d'informations disponibles via Internet dans les bases de données de biologie moléculaire (ADN, protéines...) croît de façon exponentielle. Leur étude demande de puissants moyens d'investigation.

En collaboration avec l'Unité de Recherche en Biologie Moléculaire (URBM) des Facultés Universitaires Notre-Dame de la Paix de Namur, on étudiera plus spécialement l'utilisation des réseaux de neurones pour l'exploitation des informations contenues dans les bases de données de protéines. On examinera plus spécifiquement l'applicabilité des techniques d'apprentissage automatique par un réseau de neurones dans la mise au point d'outils (matrices de scores) utilisables pour réaliser un alignement de séquences protéiques.

Abstract

Machine Learning algorithms are mainly of application in fields where important quantities of information are available. They are used to classify these data, to estimate a function, to discover relations, to simulate laws of these fields.

One will pose the theoretical bases of the main systems of artificial intelligence and of the main algorithms of learning. The limits of applicability and the validity of the induced results will be examined.

Since the sequencing of the human genome, the quantity of information available via Internet in the databases of molecular biology (DNA, proteins ...) grows in an exponential way. Their study asks for powerful means of investigation.

In collaboration with the Research Unit in Molecular biology (URBM) of University Faculties Notre-Dame de la Paix of Namur, one will study in particular the use of Neural Networks for the exploitation of the information contained in the protein databases. One will more specifically examine the applicability of the techniques of machine learning by a Neural Network in the development of tools (scoring matrices) usable to carry out an alignment of proteinic sequences.

Promoteur : Monsieur le professeur Pierre-Yves Schobbens.
Co-promoteur : Monsieur le professeur Eric Depiereux.

Avant propos

Avant de développer le contenu scientifique de ce travail de fin d'études, il me tient à cœur de remercier les personnes ayant marqué mon parcours d'une façon ou d'une autre durant ces trois dernières années.

Tout d'abord je souhaite exprimer ma profonde reconnaissance envers Monsieur le professeur Pierre-Yves Schobbens qui m'a proposé ce sujet de fin d'études, en collaboration avec le laboratoire de l'Unité de Recherche en Biologie Moléculaire des Facultés Universitaires Notre-Dame de la Paix de Namur. Je remercie Monsieur le professeur Eric Depiereux, directeur de ce laboratoire, pour m'avoir fait bénéficier de l'expérience et des connaissances acquises dans le domaine des alignements de séquences. Un des pôles de ce laboratoire est le logiciel MATCH-BOX, qui a été développé pour réaliser des alignements multiples de séquences protéiques.

Tout au long de ces trois années, j'ai éprouvé un grand intérêt à partager avec eux cette expérience de recherche de l'applicabilité de l'apprentissage automatique dans le cas précis des alignements de séquences de protéines. Je les remercie pour en avoir été les guides et de m'avoir permis de réaliser une large investigation en informatique dans le domaine de l'intelligence artificielle, et en biologie moléculaire dans l'exploitation et l'interprétation des bases de données de protéines et des bases de données des génomes. Alors qu'aucune publication scientifique n'a abordé le calcul direct des scores d'alignement d'acides aminés par un réseau de neurones, ils m'ont encouragé à continuer dans cette voie. Toute connaissance est un enrichissement, et l'exploration d'une voie nouvelle peut conduire à des conclusions intéressantes.

Je remercie tout particulièrement le Docteur Christophe Lambert, directeur du département de recherche et développement de BioXpr qui m'a fait profiter de ses conseils, de son expérience et de son « know how » de ce domaine.

Ingénieur civil, diplômé de l'Université de Liège, et informaticien de la première heure (1970), pendant vingt années, j'ai eu l'occasion de pratiquer tous les métiers de l'informatique, depuis la programmation jusqu'à la conception de systèmes, la direction de projets et l'audit informatique. C'était l'époque des mini ordinateurs, du développement « ab nihilo » de systèmes « temps réel », de la naissance des microprocesseurs, puis des ordinateurs personnels. C'est avec beaucoup de plaisir, et avec un profond intérêt que j'ai pu bénéficier d'une remise à niveau complète de mes connaissances en suivant le cursus de l'Institut d'Informatique des Facultés Universitaires Notre-Dame de la Paix de Namur. Je remercie tous les professeurs pour leur engagement et leur enseignement de qualité.

Une étape importante pour la maîtrise des aspects liés à la biologie moléculaire a été le diplôme de 3ème cycle du DES en Biotechnologie, que j'ai réalisé en parallèle et obtenu avec grade (GD) en juin 2008. Cette formation a conforté mes connaissances dans ce domaine, plus particulièrement en ce qui concerne les structures et les fonctionnalités des protéines. Je remercie tous les professeurs du Centre Universitaire de Charleroi (Académie Wallonie Bruxelles), et plus particulièrement Monsieur le professeur Jean Richelle pour ses informations sur l'analyse physique de la structure des protéines.

J'ai aussi profité d'une formation dans le domaine des processus de Markov en assistant au cours de Monsieur Davy Paindavaine de l'Institut de Statistiques de l'Université Libre de Bruxelles. J'avais également pu approfondir l'étude des processus stochastiques en suivant le cursus de troisième cycle du DES en « Gestion des risques financiers » aux Facultés Universitaires Saint-Louis (Académie Louvain), terminé avec grade (GD) en septembre 2001. Pour les réseaux de neurones Monsieur le professeur Michel Verleysen de la faculté des Sciences appliquées de l'Académie Louvain à Louvain-la-Neuve, m'a permis d'assister à son cours et aux travaux pratiques. C'est dans ce cadre que j'ai eu l'occasion de programmer des réseaux de neurones dans MATLAB, directement à partir des formules théoriques. Je le remercie, ainsi que les assistants de son service pour leurs conseils lors des travaux pratiques.

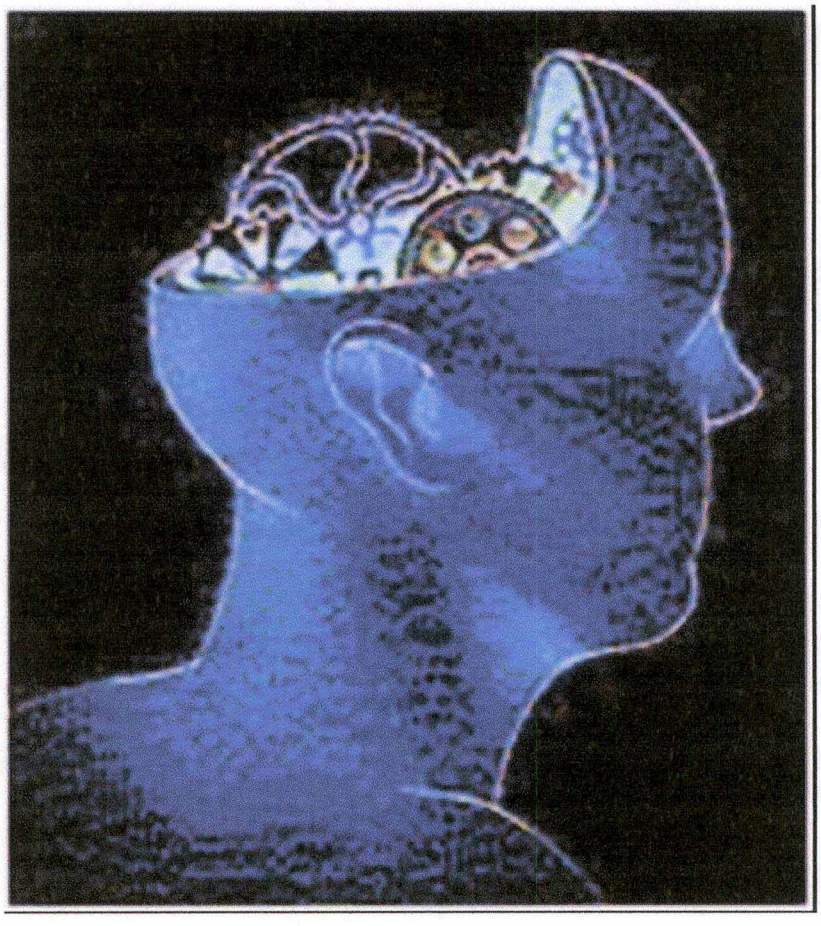
Mes remerciements vont ensuite à ma famille, qui m'a permis de concilier le temps consacré à cette étude qui me passionnait, et le temps consacré aux vacances. Il a fallu s'organiser pour mener de front la vie familiale, un travail professionnel à temps plein, et une formation universitaire à horaire décalé.

Je remercie tout particulièrement mon épouse qui m'a toujours encouragé.

Mes remerciements vont à nouveau à Christophe Lambert, pour sa relecture sur le fond et le suivi des différentes étapes de la construction de ce travail.

Table des matières

APPRENTISSAGE AUTOMATIQUE



(© Iowa State University)

1. INTRODUCTION

1.1. Objectif

L'objectif de ce travail est d'acquérir une connaissance approfondie dans le domaine de l'« Intelligence artificielle ». On réalisera une étude des principaux algorithmes, en développant plus particulièrement un aspect essentiel de ces systèmes, qui est leur faculté d'apprendre, d'extraire d'un domaine une connaissance complexe qui n'est pas facilement formalisable.

Un domaine d'étude qui présente une extrême complexité est le domaine du vivant. Les premières traces de la vie sur terre remontent à plus de trois milliards d'années. A partir d'ancêtres communs, les organismes se sont diversifiés, ainsi que les molécules qui les composent (ADN, protéines...). Le séquençage du génome de nombreux organismes a été réalisé, et de nombreux gènes sont identifiés. La transcription de l'ADN de ces organismes nous révèle l'existence d'une multitude de molécules dont nous ne connaissons que la structure chimique, mais dont nous ignorons la fonctionnalité.

Une étape importante de notre connaissance de ces organismes est d'identifier les propriétés de ces molécules en recherchant, par comparaison, des molécules « proches », de propriétés connues, dans les bases de données « biologiques ». Cette étape de recherche d'identité se fait notamment par « alignement de séquences », c'est-à-dire en comparant l'enchaînement des briques de base (les acides aminés) des molécules connues et inconnues.

Ce sera le domaine que nous allons investiguer. Comment un réseau de neurones peut-il aider à ce travail ? Convient-il pour ce type d'études? Que peut-il « apprendre », et avec quel degré de précision ?
La consultation d'ouvrages de référence, ainsi qu'une recherche bibliographique effectuée sur les sites des bibliothèques et des Instituts Scientifiques, via Internet, se sont traduit dans la rédaction de la partie théorique de ce travail. Nous avons ensuite réalisé des applications en construisant des réseaux de neurones, principalement avec le logiciel MATLAB.

1.2. Contenu du travail

Le *chapitre deux* situe l'inférence dans son contexte historique, et la présente comme un moteur pour de nombreux systèmes d'intelligence artificielle, et un élément essentiel des algorithmes d'apprentissage automatique.
Le *troisième chapitre* examine les différents algorithmes d'intelligence artificielle et explore les principaux domaines d'application potentielle de ces systèmes. Il analyse les bases théoriques de l'apprentissage automatique. Comment un ordinateur peut-il acquérir des connaissances que nous ne maîtrisons pas toujours, découvrir les lois d'un domaine que nous ne savons pas modéliser. Un point d'attention particulier sera porté sur les limites de faisabilité de ces apprentissages. On abordera le calcul de la dimension de Vapnik Chervonenkis et de l'entropie de ces systèmes.
Le *chapitre quatre* est dévolu à l'étude de systèmes particuliers : les réseaux de neurones, les modèles de Markov cachés et les réseaux bayésiens.
Le *chapitre cinq* rappellera les notions de biologie sur l'ADN et le code génétique. On examinera plus spécifiquement les protéines qui par leur nature et leur structure participent de manière cruciale au pilotage des principaux mécanismes de la vie.
Le *chapitre six* présentera la problématique des alignements de séquences. Il traitera de la comparaison des séquences d'acides aminés qui constituent les protéines et de leur alignement. Ce domaine est d'une grande importance dans la recherche des fonctionnalités d'une protéine inconnue ainsi que pour la construction de molécules à usage de catalyseurs ou de médicaments.
Le *chapitre sept* présente les expérimentations réalisées. Les réseaux de neurones seront appliqués à la construction d'une matrice de score, qui est l'outil de base pour réaliser des alignements de séquences.

2. L'INFERENCE

« Tout apprenant est un constructeur »
Piaget

L'apprentissage automatique (*machine learning*) et l'intelligence artificielle nous laissent suggérer que des machines, des ordinateurs, peuvent acquérir des modes de raisonnement et une appréhension du monde comparables à ceux de « l'Homme ». Comment une machine peut-elle apprendre à connaître et quelle intelligence peut-on lui reconnaître ?

En nous interrogeant sur les méthodes d'apprentissage qui sont les « Nôtres », sur les modes de raisonnement et les mécanismes de fonctionnement de notre cerveau, pouvons-nous définir la nature de l'intelligence que l'on attribue aux machines et mesurer sa puissance et ses limites ?

Les définitions de l'intelligence et du raisonnement peuvent-elles s'appliquer à des systèmes d'intelligence artificielle, à des machines ? [224]

> ***Raisonner*** c'est « conduire un raisonnement, enchaîner des jugements pour aboutir à une conclusion ».
> ***L'intelligence*** est « la faculté de connaître et de comprendre ».

Une machine peut-elle apprendre à découvrir son environnement et le comprendre, et ensuite après avoir acquis des connaissances pouvoir les interpréter et raisonner ?

2.1. Les philosophes antiques

Les origines connues des bases théoriques du raisonnement nous viennent des écrits que nous ont laissés les philosophes de l'antiquité. Plusieurs courants philosophiques ont posé les premiers concepts de la logique dont nous retrouvons aujourd'hui l'essence au cœur d'un certain nombre d'algorithmes d'intelligence artificielle.

Aristote (384 Ante Christus) est l'élève de Platon. Alors que Platon cherche dans le ciel des explications sur l'existence de l'homme, Aristote cherche à comprendre ce qui l'entoure à partir de la réalité des choses. Il développe sa propre vision du monde et notamment sa vision géocentrique de la terre autour de laquelle graviterait le soleil. Entre autres avancées en philosophie, il invente la « logique[1] » et sur ce plan son œuvre est révolutionnaire et a prévalu pendant quinze siècles. Selon Aristote, la logique a pour but de fonder une connaissance rigoureuse du monde.

Ses écrits de logique sont rassemblés dans « l'Organon » [71] [22]. Dans ce traité il pose les bases de la déduction logique, appelée par lui « syllogisme ».

> *« Le syllogisme est un discours dans lequel certaines choses étant posées, quelque chose d'autre que ces données en résulte nécessairement par le seul fait de ces données. »*
>
> *Aristote, Premiers Analytiques, I 1 24 a 10*

Dans la logique aristotélicienne, un syllogisme est un raisonnement basé sur deux propositions supposées vraies, les prémisses : une prémisse majeure « Tous les Hommes[2] sont nés sur la Terre » et une prémisse

[1] le terme logique est apparu plus tard et n'était pas utilisé par Aristote
[2] Homme, synonyme de « Etre humain »

mineure « Tous les informaticiens sont des Hommes ». Le terme majeur (sont nés sur la terre) et le terme mineur (les informaticiens) sont reliés par un terme moyen (les Hommes) dans la conclusion : « Tous les informaticiens sont nés sur la terre ».

Il faut que les propositions soient vraies: jusqu'à présent aucun Homme n'est né dans l'espace.
Il faut aussi être attentif à la définition sémantique des deux propositions pour éviter d'arriver à un paradoxe.

> *Plus il y a de voyageurs et moins il y a de places dans le train*[3].
> *Plus il y a de places dans le train*[4] *et plus il y a de voyageurs.*
> *Plus il y a de places dans le train et moins il y a de places dans le train.*

La logique aristotélicienne n'est pas que déductive. Il définit également le syllogisme inductif qui est le passage du particulier au général.

> *Toute conviction s'acquière par le syllogisme ou provient de l'induction*
> *Aristote, Premiers Analytiques, II 23 68 b 13*

L'exemple donné par Aristote est le suivant :

> *L'âne, le mulet, le cheval vivent longtemps ;*
> *Or, ce sont là tous des animaux sans fiel;*
> *Donc, tous les animaux sans fiel vivent longtemps.*

C'est un raisonnement hypothétique qui tente d'expliquer le général par des observations sur le particulier. Il repose sur des suppositions.[5]

L'abduction est, selon Aristote, un syllogisme, dont la majeure est certaine et la mineure seulement probable [22]. Il en résulte que la conclusion, sans être certaine comme la majeure, acquiert la probabilité de la mineure.

> *La science peut être enseignée (majeure certaine); la justice est une science (mineure probable); donc la justice peut être enseignée...*
> *ARISTOTE* (Larousse encyclopédique).

Une autre avancée va être faite par l'école philosophique des « Stoïciens ». Ils vont établir les bases de la logique des propositions, sans pour autant supplanter la logique aristotélicienne.

> *Une proposition vraie est ce qui est, et une proposition fausse est ce qui n'est pas.*
> Sextus Empiricus , Contre les professeurs, VIII, 84)

Les Stoïciens vont critiquer la logique aristotélicienne. Sextus Empiricus, va mettre en doute les règles d'induction car pour lui un raisonnement universel ne peut pas se baser sur des affirmations incomplètes, vraies seulement pour un nombre limité de cas particuliers. Si on ne prend en compte qu'une partie des observations possibles sur les cas particuliers, alors l'induction ne sera pas fiable car une partie des cas qui ont été omis dans l'induction pourra contredire le cas général. De plus, si on essaye de prendre tous les cas particuliers possibles, on pourrait se trouver devant une solution impossible car les cas particuliers peuvent être infinis, ou non définis.

---

[3] dans les wagons de ce train
[4] plus il y a de wagons dans ce train
[5] hypothèse : «ce sont là tous des animaux sans fiel »

2.2. Epistémologie

Au cours des siècles, pédagogues et philosophes ont essayé de comprendre et de définir ces mécanismes qui permettent à l'homme d'apprendre, de comprendre, d'acquérir et de mémoriser des connaissances. Ils définissent l'apprentissage par deux approches, soit sur base du résultat c'est à dire de la connaissance qui est acquise, soit sur base du processus c'est à dire des mécanismes mis en œuvre pour acquérir des connaissances.

Les trois principaux courants de pensée sont les suivants :

Conception behavioriste :
Elle s'intéresse principalement à la connaissance acquise. C'est un apprentissage par approximations successives : à force de répéter la même configuration d'informations on « enregistre » le résultat, on acquière une connaissance. C'est une explication physiologique de l'apprentissage, basée sur le conditionnement. L'apprentissage est vu comme un réflexe conditionnel (Watson, Pavlov, Skinner).

Conception constructiviste :
C'est un mécanisme qui sur base d'un état de connaissances déjà acquises et sur réception de nouvelles informations, restructure la connaissance passée. Les concepts ne s'enseignent pas, ils se construisent au cours de stades d'évolution successifs grâce à l'interaction de l'individu avec son environnement. Les sollicitations et contraintes de l'environnement vont progressivement modifier l'équilibre de l'acquit pour le faire évoluer (Piaget).

Conception cognitiviste :
Le cognitivisme est un courant de pensée de la psychologie contemporaine qui s'interroge sur la genèse de la connaissance. Pour le cognitivisme la pensée est un processus de traitement de l'information. C'est une approche pour comprendre les processus d'apprentissage et de structuration de la connaissance. On considère la manière dont chaque individu pense et agit, ce qui oriente sa perception et ses jugements et l'amène à créer sa vision personnelle du monde (Gordon Allport).

La « cognition » est l'ensemble des grandes fonctions permettant à l'organisme d'interagir avec le milieu (perception, mémoire, intelligence, ...). Si les sciences cognitives ont pour objectif d'expliquer les mécanismes qui sont à la base de la connaissance humaine, dans quelle mesure l' « Intelligence artificielle » peut-elle la simuler et la reproduire ? La « Cognition » est-elle la base des fondements théoriques de l' « Intelligence artificielle » ?

2.3. Vers la logique moderne

Pour que la logique fasse de grands progrès, il faudra attendre l'apport de Gottfried Wilhelm von Leibniz (1646-1716) et son travail sur la constitution d'un langage universel et sur les bases de la formalisation algorithmique et mathématique de la logique. Leibniz a en particulier introduit une grande partie de la notation mathématique moderne (usage des quantificateurs, symbole d'intégration...) [149].

La logique mathématique moderne est née à la fin du XIX^{ème} siècle avec des mathématiciens comme Frege, Boole, Hilbert [72].

Sur le plan philosophique, l'induction reste un aspect de la logique toujours discuté par les philosophes (Karl Popper , Wesley Salmon) [98].

L'apprentissage automatique a été formalisé par Valiant, Vapnik, ... qui lui ont donné une base théorique. Nous développerons cette approche dans le chapitre trois.

2.4. De l'intelligence de l'Homme à l'intelligence artificielle [6]

Comment une machine peut-elle « imiter » l'intelligence de l'homme pour raisonner, parler, calculer, apprendre, ...? Comment s'y prendre pour créer une ou de l'intelligence artificielle ? Deux types d'approches ont été essentiellement explorés:

- soit, procéder d'abord à l'analyse logique des tâches relevant de la cognition humaine et tenter de les reconstituer par programme. C'est cette approche qui a été privilégiée par l'Intelligence Artificielle et la psychologie cognitive classique. Cette démarche est étiquetée sous le nom de *cognitivisme* ;
- soit, puisque la pensée est produite par le cerveau ou en est une propriété, commencer par étudier comment celui-ci fonctionne. C'est cette approche qui a conduit à l'étude de réseaux de neurones formels. On désigne par *connexionnisme* la démarche consistant à vouloir rendre compte de la cognition humaine par des réseaux de neurones.

La seconde approche a donc mené à la définition et l'étude de réseaux de neurones formels qui sont des réseaux complexes d'unités de calcul élémentaire interconnectées.

2.5. La physiologie du cerveau

Un bref aperçu de quelques propriétés élémentaires de neurophysiologie vont permettent de relier neurones réels et neurones formels. Le cerveau est constitué de milliards de cellules (les neurones) qui sont interconnectées entre elles. Les neurones reçoivent les signaux (impulsions électriques) par des extensions très ramifiées de leur corps cellulaire (les dendrites) et envoient l'information par de longs prolongements (les *axones*) vers des neurones voisins. La durée de chaque impulsion est de l'ordre d'une milliseconde et son amplitude d'environ cent millivolts.

Les contacts entre deux neurones, de l'axone à une dendrite, se font par l'intermédiaire des synapses. Lorsqu'un potentiel d'action atteint la terminaison d'un axone, des neuromédiateurs sont libérés et se lient à des récepteurs post-synaptiques présents sur les dendrites. Le signal passe ainsi du neurone émetteur au neurone récepteur. Chaque neurone peut gérer en permanence jusqu'à un millier de signaux. Ces signaux ne sont pas transmis si le potentiel d'action ne dépasse pas un seuil déterminé au contact synaptique.

Quelques informations en vrac :

- le cerveau contient environ 100 milliards de neurones.
- la vitesse de propagation des influx nerveux est de l'ordre de 100 m/s. C'est à dire bien inférieure à la vitesse de transmission de l'information dans un circuit électronique.
- on compte de quelques centaines à plusieurs dizaines de milliers de contacts synaptiques par neurone. Le nombre total de connexions est estimé à environ 10^{15}.
- la connectique du cerveau ne peut pas être codée dans un « document biologique » tel l'ADN pour de simples raisons combinatoires. La structure du cerveau provient donc en partie des contacts avec l'environnement. L'apprentissage est donc indispensable à son développement.
- on observe par contre une grande plasticité de l'axone, des dendrites et des contacts synaptiques. Celle-ci est surtout très importante après la naissance. Cette plasticité est conservée tout au long de l'existence.

Il semble que l'apprentissage se fasse par un double mécanisme: des connections sont établies de manière redondantes et aléatoires puis seules les connexions entre des neurones simultanément actifs sont conservées (phase de sélection) tandis que les autres sont éliminées. On parle de stabilisation sélective.

[6] Groupe de Recherche sur l'Apprentissage Automatique - Université de Lille 3.

3. L'APPRENTISSAGE

3.1. Le processus d'apprentissage

3.1.1. Connaissance et intelligence

« Apprendre c'est acquérir des connaissances » (Larousse), c'est aussi un préalable à l'expression de l'intelligence, qui est l'aptitude à comprendre. Apprendre, c'est découvrir et reconnaître les éléments qui constituent son environnement, ainsi que les lois ou les théories qui expliquent son fonctionnement. L'intelligence c'est maîtriser ces connaissances acquises et les utiliser dans des processus de raisonnement.

L'apprentissage d'un système va être son aptitude à acquérir de nouveaux comportements à partir de ses interactions avec l'environnement [20]. C'est un processus dynamique, où l'information reçue va être interprétée, modélisée, mémorisée et disponible, soit pour répondre à une sollicitation extérieure, soit pour générer des actions.

Les différents algorithmes d'intelligence artificielle ont en commun un sous-système essentiel, celui d'acquérir des connaissances et l'expertise dont ils ont besoin.

Pour construire ces systèmes, une première approche est de mettre à leur disposition une base complète de connaissance qui contient toutes les règles de logiques qu'ils devront appliquer. Ceci peut être réalisé lorsqu'on connaît a priori de manière précise et exacte les traitements qui devront être effectués. La plupart des systèmes à base de règles se retrouvent dans cette catégorie.

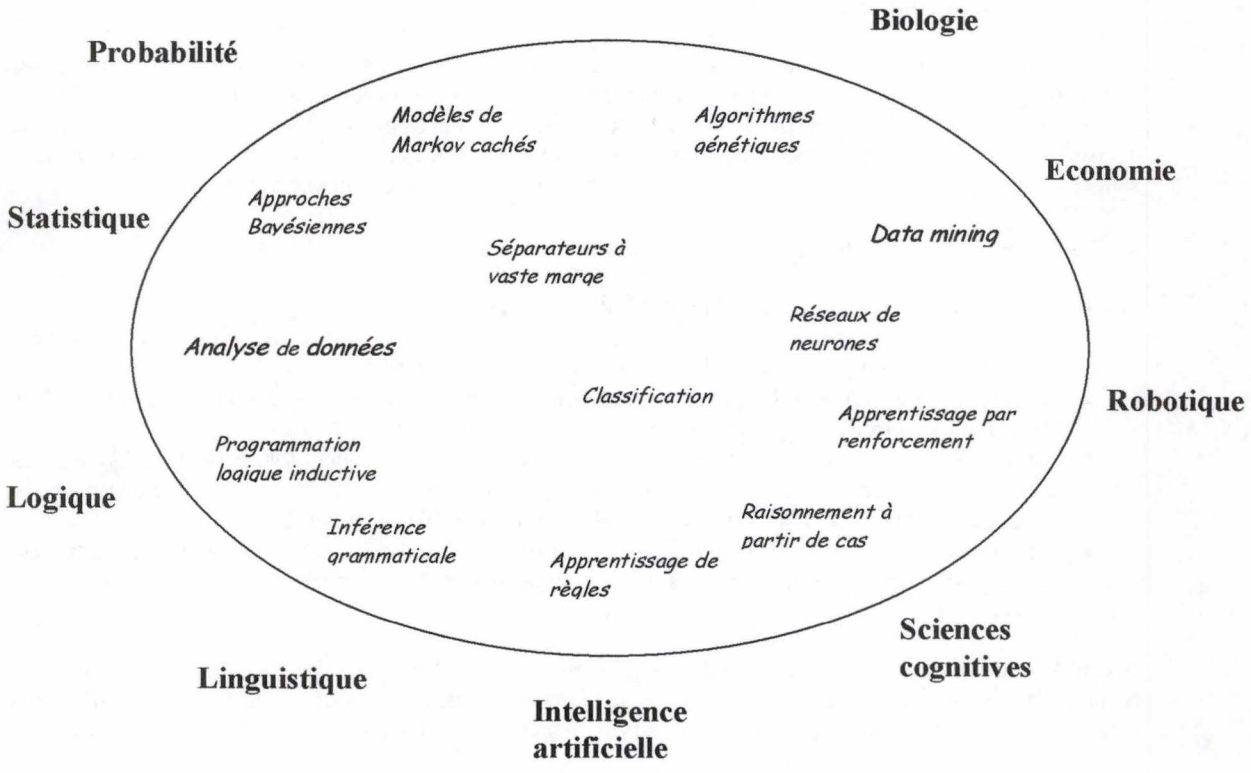


Figure 1 - Cartographie des algorithmes d'intelligence artificielle

« Or, lorsqu'on travaille sur des domaines complexes et mal formalisés ..., on se trouve ainsi amené à concevoir et développer des « outils apprenants » susceptibles d'analyser de manière autonome les données issues de l'environnement et, sur cette base, aptes à prendre des décisions, voire à élaborer des stratégies d'interactions. »[7] [106].

On se situe alors dans le domaine de l'apprentissage automatique ou « *Machine learning* » [5].

La carte ci-dessus donne un aperçu des différentes techniques ou algorithmes d'apprentissage. Elle situe ce domaine à la confluence de nombreuses disciplines, qui sont génératrices ou utilisatrices de ces méthodes [184].

3.1.2. Acquisition de connaissances

Pour être opérationnel, un système d'intelligence artificielle [221] doit donc posséder des informations adéquates sur son domaine spécifique de fonctionnement. La formalisation de ces connaissances, leur acquisition par le système, les protocoles qui sont mis en place pour créer des bases de connaissance, sont des étapes essentielles et critiques qui conditionneront le bon fonctionnement du système d'intelligence artificielle.

La connaissance du domaine : cette connaissance peut exister sous diverses formes.

- Pour la recherche scientifique ou la médecine, cette connaissance peut être celle de quelques experts isolés qui sont les seuls à maîtriser un domaine de pointe très particulier.
- En astronomie, le mouvement des corps dans l'espace peut être décrit par des lois théoriques.
- Pour la découverte d'un environnement inconnu, non formalisé, sans lois théoriques exprimables, la connaissance peut se traduire sous forme d'une série de résultats d'expérience. C'est par exemple le cas de la conduite automatique de robots ou de véhicules.
- En économie ou sur les marchés financiers, la connaissance peut être la traduction d'une collection d'observations, en complément des modèles théoriques.

Le transfert de connaissances : l'extraction de la connaissance du domaine, sa mise en forme et la création de base de données de connaissances sont les étapes de l'apprentissage du système. Il y a deux approches possibles.

- Un spécialiste et un expert s'associent pour créer entièrement un modèle de toute la connaissance du domaine et constituer la base de connaissance.
- Un algorithme d'apprentissage enregistre automatiquement un ensemble de connaissances codifiées, et les traduit lui-même dans un formalisme interne qui est l'état de son savoir.

L'exploitation des connaissances par les systèmes d'intelligence artificielle : on distingue deux grandes catégories, les systèmes à base de règles ou systèmes symboliques (systèmes experts, algorithmes génétiques, ...) et les systèmes connexionnistes (réseaux de neurones, machines à vecteurs de support,...). Ces deux grands types de systèmes peuvent également collaborer pour former des systèmes mixtes.

[7] (ref : http://www-leibniz.imag.fr/Apprentissage/index.html)

La figure ci-après montre le parallélisme et les interrelations entre ces deux approches.

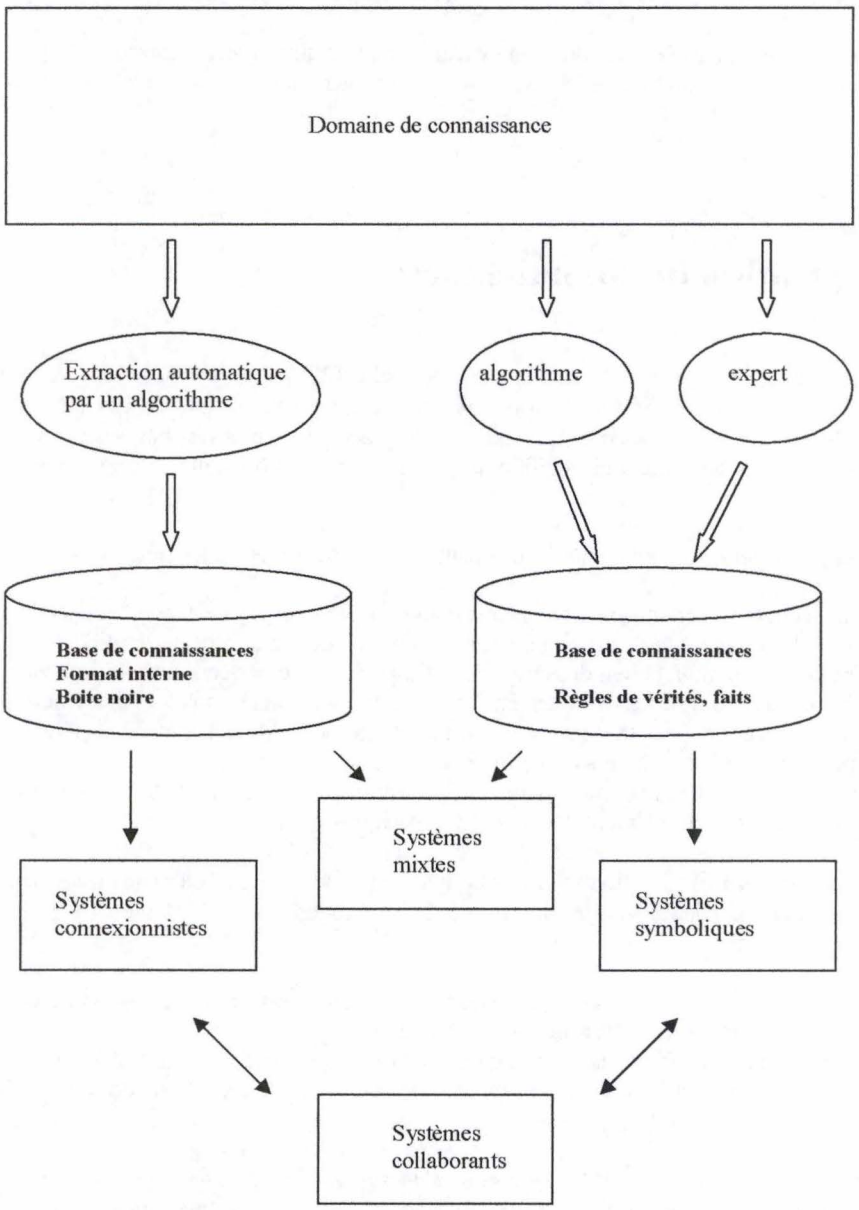


Figure 2 - Parallélisme entre systèmes symboliques et connexionnistes

3.1.2.1. *Approche symbolique*

C'est la voie qui conduit à la réalisation de systèmes experts [149] [234].

Apprentissage par un expert

Un spécialiste du domaine étudié va construire un modèle, en décrivant toutes les règles qui régissent le fonctionnement de ce domaine sous forme de propositions et constituer la base de connaissance spécifique à ce domaine. Le système expert se base sur la logique des propositions. Il dispose d'un moteur d'inférence qui lui permet d'automatiser les aspects du raisonnement humain. Sa logique interne va lui permettre de répondre à des questions sur le domaine, dans les limites de la base de connaissance mise à sa disposition.

Apprentissage automatique

La base de règles peut également être construite de manière automatique. Les éléments de connaissance du domaine sont alors traités par un algorithme qui va créer les règles de la base de connaissance.

Les principales méthodes sont [56] [146]:

- Les arbres de décision :

Les arbres de décision sont des classifieurs pour des instances représentées par une liste de critères attributs/valeur.

- Les nœuds de l'arbre testent les attributs.
- Il y a une branche pour chaque valeur de l'attribut testé.
- Les feuilles spécifient les catégories (deux ou plus).

ID3 (*Iterative Dichotomies 3*) construit un arbre de décision à partir d'un ensemble fixe d'exemples. Un exemple a plusieurs attributs et appartient à une classe (comme oui / non). Ce sont des attributs uniquement nominaux. Les nœuds feuilles de l'arbre de décision contiennent le nom de la classe, alors que les nœuds intermédiaires sont des « centres de décision ». A chacun de ces nœuds, on teste un attribut, qui oriente le choix d'une des branches sous jacentes (Quinlan 1986). L'algorithme essaye de réaliser une discrimination parfaite de tous les exemples, ce qui nécessite de réaliser des élagages de branches (*pruning*), ou un arrêt prématuré de l'apprentissage lorsque les exemples sont bruités (Quinlan). ID4 et ID5 sont des évolutions de ID3 pour le rendre incrémental et accepter une suite continue d'exemples [172].

CART (*Classification And Regression Trees*) est un arbre de décision, qui n'accepte que des choix binaires à chacun de ses nœuds. Le principe de l'algorithme est de rechercher la partition la plus efficace à chaque nœud, sur base de l'ensemble des exemples d'apprentissage. Il favorise la partition la plus équilibrée possible en nombre d'exemples, et qui permet d'atteindre rapidement un nœud feuille [146].

C4.5 est une évolution de l'algorithme ID3 de Quinlan. Il n'est pas binaire comme CART et permet de créer un nombre important de branches à chaque nœud de décision, ce qui peut le rendre touffu. L'algorithme est basé sur la recherche de la croissance maximum de gain d'information à chaque nœud de décision.

Il comporte deux phases.

- Une phase d'expansion, où on construit récursivement l'arbre de décision, en divisant l'ensemble d'apprentissage.
- Une phase d'élagage, basée sur un heuristique qui élimine les branches non significatives (erreurs, exemples mal classés).

C5 et See5 sont les versions actuellement commercialisées par Quinlan. [8]

- Les algorithmes génétiques d'induction de règles [18] [193] [208].

Les algorithmes génétiques sont une métaphore de l'évolution du vivant, car ils simulent une succession de génération d'êtres qui évoluent dans le temps. L'apprentissage est représenté comme une compétition à l'intérieur d'une population de candidats potentiels à la solution du problème (appelés chromosomes). A chaque étape (une génération) une fonction d'adéquation (*fitness function*) évalue chaque solution (hypothèse) et décide si elle va être retenue pour participer à la génération suivante, sur base d'un critère prédéfini. Ensuite, par des opérations qui miment les modifications génétiques (croisements, mutations), l'algorithme va créer une nouvelle population de candidats à la solution.
Les algorithmes génétiques se révèlent efficaces pour la recherche et l'apprentissage de règles par induction à partir d'une base d'exemples.

- Le raisonnement fondé sur des cas

Les systèmes de type CBR (*Case Based Reasonning*) simulent le raisonnement de l'expert humain, qui fait appel à sa connaissance, ses expériences vécues pour les adapter à une nouvelle situation et résoudre un nouveau problème [34] [121].

3.1.2.2. *Approche connexionniste*

Apprentissage automatique adaptatif

L'apprentissage est un processus graduel, itératif où des paramètres (poids d'un réseau de neurones) sont modifiés un nombre très important de fois, pour tendre vers une valeur cible qui concrétisera l'état de connaissance acquis par le système [40] [68].

Les méthodes d'apprentissage peuvent être réparties en trois grandes classes, selon le degré de contrôle que l'on garde sur le processus d'acquisition des connaissances :

- Apprentissage supervisé

L'apprentissage supervisé est une classification. On dispose de « points » appartenant à n classes différentes : c'est l'ensemble d'apprentissage. Ce dernier va permettre de construire un modèle qui va définir des frontières dans l'espace de description. L'apprentissage va construire les paramètres qui vont définir les limites des classes. Ensuite, lorsque l'on soumet un nouvel élément « non classé » au modèle, celui-ci va définir sa classe d'appartenance, sur base de son espace descriptif.

[8] www.RuleQuest.com

- Apprentissage semi supervisé

L'apprentissage semi supervisé est une méthode d'apprentissage qui est capable d'apprendre à partir de données incomplètement décrites. Elle se base sur un ensemble d'apprentissage, dont on ne connaît la classe d'appartenance que pour une fraction de ses éléments constitutifs. On a donc besoin d'informations supplémentaires entre les éléments de l'ensemble d'apprentissage, pour associer les éléments non classés, à des groupes d'éléments classés. On définit ainsi pour les éléments non classés au départ, leur appartenance à une classe.

- Apprentissage non supervisé

L'apprentissage non supervisé est un processus de regroupement d'éléments qui ne sont caractérisés par aucun paramètre de classification. Le processus construit des groupes d'éléments similaires (*cluster*) sur base des propriétés qui caractérisent ces éléments. Ces groupes ou « *cluster* » qui se sont dégagés de l'ensemble des éléments vont constituer les classes du modèle. Pour les nouveaux éléments qui apparaissent, on va en déduire leur groupe d'appartenance, sur base de leurs caractéristiques propres.

3.1.3. La validation des connaissances

C'est l'étape qui suit la phase d'apprentissage et pendant laquelle on effectue une vérification des connaissances acquises par le système. Les données disponibles, décrivant le système ont été fractionnées en deux sous-ensembles. La première partie est utilisée pour réaliser l'apprentissage et la seconde partie est un ensemble de test qui va servir à interroger le système, lorsque l'apprentissage est réalisé, et vérifier la validité de ses réponses.

La technique du « *bootstrap* », consiste à faire des échantillonnages successifs dans l'ensemble servant à l'apprentissage. La technique du « *jacknife* » est un partitionnement de l'ensemble de test en n parts, dont n-1 servent à l'apprentissage et dont la dernière est utilisée pour valider les connaissances.

3.2. Les méthodes de raisonnement

Elles font partie des connaissances du système.

3.2.1. La logique

Inférence : ce sont des règles logiques qui permettent de tirer des conclusions à partir de connaissances acquises (mémorisées) et de règles ou de relations entre ces connaissances.

3.2.2. Les méthodes statistiques ou probabilistes

Inférence statistique : à partir de caractéristiques individuelles, observées sur les éléments d'un échantillon tiré de manière aléatoire dans une population, on détermine les paramètres de cet échantillon. Si cet échantillon est représentatif, on extrapole les valeurs calculées pour l'échantillon pour estimer les valeurs de la population. Le tirage de n échantillons permet de calculer un intervalle de confiance [26].

3.3. L'apprentissage automatique

3.3.1. Formalisation

Les techniques et algorithmes d'apprentissage sont fort différents dans leur conception. Ils concourent cependant à un même but, celui de construire un modèle capable de réaliser un apprentissage dans un « environnement particulier » et d'exploiter les connaissances acquises [38] [132] [172].

La modélisation mathématique du problème de l'apprentissage va tenter de leur donner une base commune.

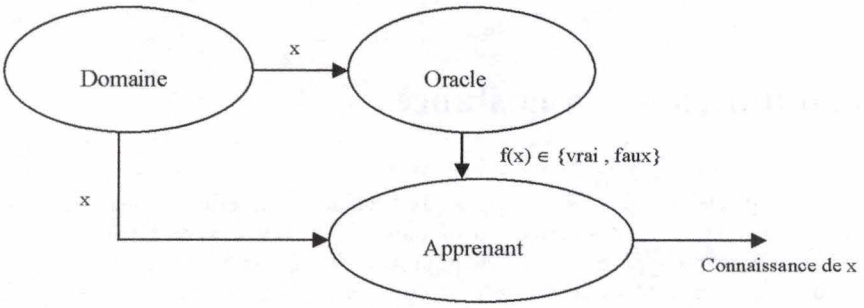


Figure 3 - Apprentissage supervisé par un "Oracle"

Le trois éléments principaux d'un système sont :

- l'algorithme d'apprentissage ou « machine apprenante », qui va recevoir des informations du domaine. Il les enregistre en phase d'apprentissage et donne un message de réponse en phase d'interrogation ;
- le domaine, qui peut être extrêmement varié: ensemble de formes à reconnaître, étude d'un environnement inconnu, reconnaissance du langage, de l'écriture, recherche d'une fonction, ... Le domaine représente un environnement complexe ;
- la source d'information, qui va fournir à la machine apprenante des messages positifs et/ou négatifs sur la vérité du domaine. On parle d'oracles qui sont censés donner une information exacte, ou d'experts, qui expriment leur opinion. L'information reçue par l'algorithme d'apprentissage pourra être incomplète et entachée de bruit ou d'erreurs.

En apprentissage supervisé, l'apprenant reçoit une information effective de l'oracle. En apprentissage non supervisé, il doit interpréter seul les informations reçues du domaine.

3.3.2. Modèle théorique

Définition des constituants du modèle

Σ	Le domaine d'apprentissage est constitué par un ensemble d'éléments de base, représentés par un alphabet « sigma » Σ
Σ^*	L'ensemble des instances ou objets qui peuvent appartenir au domaine est Σ^*, l'ensemble de toutes les combinaisons des éléments de l'alphabet
D	Le domaine d'apprentissage, qui est constitué d'éléments de Σ^*
X	C'est un sous-ensemble de Σ^*, contenu dans D. Un exemple particulier $x : x \in X$, l'espace des exemples.
C	C'est l'ensemble des concepts sur Σ qui peuvent être définis dans le domaine D. Un concept est une loi, une propriété, une caractéristique de groupes d'éléments du domaine
c	Un concept particulier est une fonction qui appartient à l'espace des concepts ($c \in C$) Lorsque cette fonction est appliquée à un espace d'exemples, elle détermine si l'exemple répond ou non au concept $c : x \rightarrow \{0,1\}$ ➢ l'exemple est positif si $c(x) = 1$ ➢ l'exemple est négatif si $c(x) = 0$
H	L'objectif du processus d'apprentissage est de construire un concept d'hypothèses h qui correspond au concept cible c que l'on étudie $h \in H$, l'espace des hypothèses sur Σ
Err	L'erreur est définie par la probabilité que h soit différent de c $Err = p \{ x \in X \mid h(x) \neq c(x) \}$

Un algorithme d'apprentissage $\mathcal{A}$ pour (C, H) est une procédure qui accepte en entrée des échantillons d'exemples pour des fonctions de C et qui donne en sortie des hypothèses correspondantes dans H.

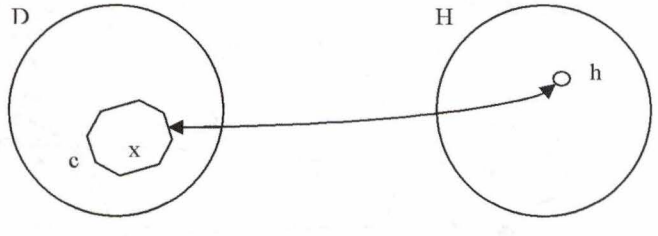


Figure 4 - Domaine d'apprentissage et domaine d'hypothèses

Après apprentissage, le système interrogé sur l'appartenance d'un exemple à un concept, va sélectionner dans son espace d'hypothèses, une hypothèse h. Un apprentissage correct doit générer une réponse

$$h: h(x) = c(x).$$

Illustrons ce modèle par un exemple. Considérons comme alphabet les symboles du jeu de cartes et définissons les constituants du modèle dans ce cas particulier.

Alphabet $\Sigma = \{\blacklozenge, \clubsuit, \spadesuit, \heartsuit\}$
Domaine $D = \{\clubsuit, \clubsuit, \spadesuit, \spadesuit, \spadesuit, \spadesuit\}$

Concepts $C = \{\clubsuit = \text{trèfle}, \spadesuit = \text{pique}\}$
Hypothèses $H = \{\text{pique, cœur, trèfle, carreau}\}$

Apprentissage des concepts sur la série d'exemples X_1 et vérification de l'apprentissage sur la série d'exemples X_2 :

Echantillon d'exemples $X_1 = \{\clubsuit, \spadesuit, \spadesuit\}$	Echantillon d'exemples $X_2 = \{\clubsuit, \clubsuit, \spadesuit\}$
Selon le concept étudié :	
Trèfle(♣)=1, Pique(♣)=0	h $[X_2(1)]$ = Trèfle
Trèfle(♠)=0, Pique(♠)=1	h $[X_2(2)]$ = Trèfle
Trèfle(♠)=0, Pique(♠)=1	h $[X_2(3)]$ = Pique

Représentation des espaces

Si un système est susceptible d'apprendre, se pose la question de définir quelles sont les fonctions qui peuvent être apprises, quelle est la complexité d'un algorithme d'apprentissage.

Les travaux de Gold, Valiant et Vapnik posent des premières bases théoriques. Quel va être pour un système sa faculté d'apprendre, et avec quelle marge erreur?

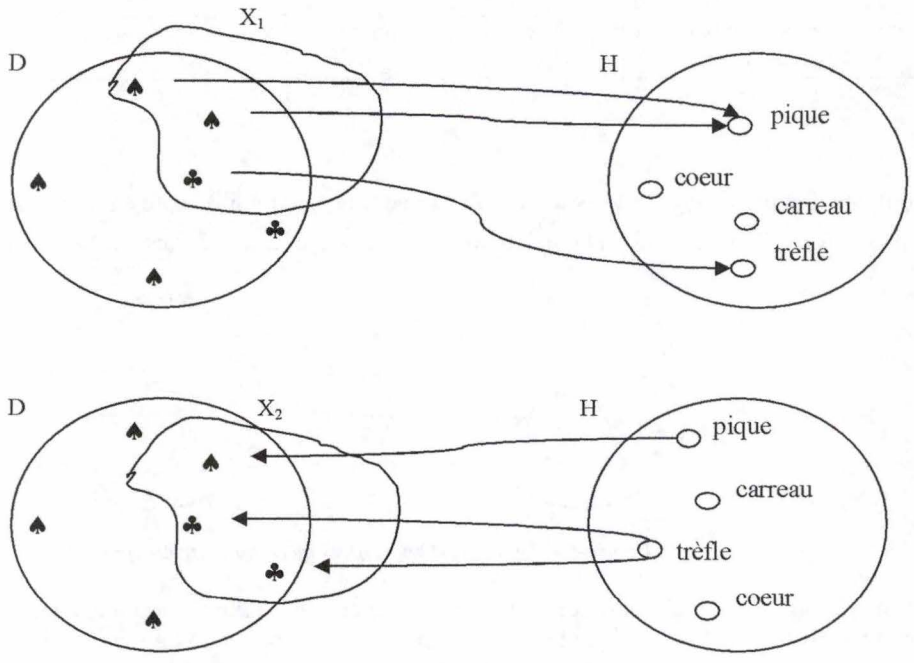


Figure 5 - Apprentissage d'un concept

3.4. Paradigme de Gold (1967)

Gold définit un modèle d'apprentissage, qui permet de déterminer si un concept « c » est apprenable à la limite.

- Soit « c » = $\{x_1, x_2, \ldots, x_n, \ldots\}$, un concept qui est représenté par un nombre infini d'exemples.
- Soit « F » = $\{x_1, x_2, \ldots, x_p\}$, un flux d'exemples, qui est un sous-ensemble ordonné de « c » . Le flux est une énumération d'exemples et de contre-exemples.

A chaque nouvel exemple x_j reçu, l'apprenant va générer une hypothèse $h(x_j)$ en réexaminant l'hypothèse précédente $h(x_{j-1})$ sur base de l'information contenue dans x_j . Le flux « F » va produire $\{ h(x_1), h(x_2), \ldots, h(x_p) \}$, un flux d'hypothèses reformulées à chaque exemple pour tendre vers le concept « c ».

Gold énonce que si le flux « F » est soumis à un algorithme d'apprentissage « A » qui fournit en sortie un algorithme « G », dont on va déduire un flux d'hypothèses.

- Le concept « c » est « apprenable » à la limite, si il existe un flux F d'exemples x_i inclus dans « c » tel qu'il existe un algorithme d'apprentissage A qui construise un algorithme G qui reconnaisse « c » , c'est à dire qui génère un flux d'hypothèses qui convergent vers « c » :

$$\exists\, h(x_p) : [h(x_p) = h(x_{p+1}) = h(x_{p+n})] \wedge [(\, h(x_p) = \text{« c »}] \,.$$

- Ce modèle d'apprentissage à la limite donne une mesure de la complexité de l'algorithme d'apprentissage « A », en fonction de la cardinalité de | « F »| .

Convergence

Gold démontre qu'un certain nombre de fonctions sont identifiables à la limite. Ce sont notamment les fonctions qui sont construites à partir de polynômes d'ordre croissant. Lorsque le polynôme construit sur flux d'apprentissage est d'ordre supérieur au polynôme du concept à découvrir, le nombre d'équations dépasse le nombre d'inconnues et la solution est induite.

3.5. Paradigme de Valiant (1984)

Valiant développe la notion d'algorithme PAC : *« Probablement approximativement correct »[169] [237].*

Considérons le modèle théorique, auquel on va ajouter des facteurs d'incertitude :

X	est l'espace des instances x sur un domaine (Σ^*)
c	est un concept particulier du domaine
$c \subseteq X$	le concept c est un concept défini sur X
$x \in c$	l'exemple x appartient à la classe du concept c
C	classe des concepts possibles sur X: $C : \{c \mid c \subseteq X\}$
t	un concept inconnu à apprendre $t \in C$
P	$\{0,1\}^n$ la distribution de probabilité sur X, pour que $x_i \in t$ (arbitraire, stationnaire dans le temps, inconnue de l'apprenant)

Le flux d'informations :

On fournit à l'algorithme d'apprentissage une séquence d'exemples :

$(x_0,v_0), (x_1,v_1), (x_2,v_2), \ldots, (x_i,v_i), \ldots, (x_k,v_k), \ldots$

avec $x_i \in X$

v_i = « + » si $x_i \in t$

v_i = « - » si $x_i \notin t$

En général l'algorithme apprend avec des cas positifs et négatifs. On pourrait le faire apprendre avec des cas uniquement positifs, ou uniquement négatifs.

La probabilité d'occurrence de $x_i \in t$ dans la séquence est de $p(x_i)$. Le concept construit par l'algorithme est h. Le concept t, représenté par h, est connu de manière approximative, avec une erreur résiduelle $E = \{ x \mid h(x) \neq t(x) \}$. Le taux d'erreur τ^E (de réponses incorrectes) est

$$\tau^E = \sum_{x \in E} p(x_i) \qquad \text{avec} \qquad 0 < \tau^E < 1$$

Les facteurs d'incertitude :

Valiant définit les paramètres suivants, qui devront être respectés par l'algorithme.

- Le facteur de précision ε ($0 < \varepsilon < 1$): le taux de réponses incorrectes de l'algorithme après apprentissage doit être inférieur à $< \varepsilon$
- Le facteur de confiance δ ($0 < \delta < 1$): la probabilité pour que la contrainte sur le facteur de précision ε soit respectée est de $1 - \delta$
-

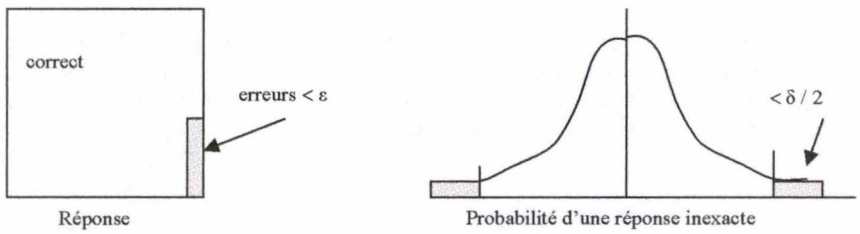


Figure 6 - Intervalle de confiance

L'algorithme d'apprentissage PAC :

Après avoir traité une séquence finie d'exemples, l'algorithme génère en sortie un concept h, représentatif du concept t. Le niveau d'apprentissage de l'algorithme est croissant avec le nombre d'exemples traités.

Si on définit la différence entre les deux concepts par $h \Delta t$, la fonction suivante Prob ($h \Delta t$) définira le taux d'erreur sur la classe de concept, c'est à dire la probabilité pour que t et h classent différemment un exemple tiré de manière aléatoire. Cette probabilité doit être inférieure à ε pour que le résultat soit probablement correct.

L'algorithme d'apprentissage est « probablement approximativement correct » PAC, si la probabilité que h donne un résultat probablement correct est au moins de $1 - \delta$. On dira que l'algorithme apprend de manière PAC la classe de concept C.

Généralisation

Si on généralise à un espace d'instances de dimension n, l'algorithme d'apprentissage A est PAC pour une classe de concepts C dans l'espace de représentation H, si :

1. A prend comme entrées ε, δ, n.
2. A peut appeler l'oracle qui lui retourne des exemples $c \in C$, selon la distribution de probabilité Prob.
3. Pour tout concept $c \in C$ et toute distribution de probabilité Prob, l'algorithme d'apprentissage A génère une réponse h tel que l'on ait la probabilité $1 - \delta$ pour que le taux d'une mauvaise identification soit $Prob(c \Delta h) \leq \varepsilon$. Soit : $Prob \{ Prob(c \Delta h) \leq \varepsilon \} \geq \delta$.

Définition :

La classe C est apprenable en un temps polynomial s'il existe un algorithme déterministe pour C dans R avec une complexité temporelle $t(\varepsilon, \delta, n, m)$ bornée par un polynôme de type :

$$Poly(1/\varepsilon, 1/\delta, n, m)$$

La complexité des algorithmes PAC

Toutes les classes ne peuvent pas être apprises par un algorithme PAC, soit en raison de la complexité des calculs (temps infini, mémoire excessive), soit en raison de la complexité de l'information qui est requise pour réaliser l'apprentissage (nombre infini d'exemples) [9] [10].

Convergence de l'algorithme PAC

Considérons des hypothèses h_1, h_2, h_k de l'espace fini des hypothèses H, qui sont les résultats successifs de l'apprentissage d'un concept objectif t.
L'algorithme d'apprentissage est PAC, donc il vérifie la règle : $prob(\text{ erreur } (h,t) > \varepsilon) < \delta$.

La probabilité que la première hypothèse h_1 classe correctement un exemple est au plus $(1 - \varepsilon)$.
Et pour que h_1 classe correctement m exemples, la probabilité est $\leq (1 - \varepsilon)^m$. De même une deuxième hypothèse h_2 aura également une probabilité maximum de $(1 - \varepsilon)^m$ de classer correctement m exemples. Comme on ne conserve qu'une hypothèse, cette dernière aura une probabilité : $P(h_1 \text{ ou } h_2) \leq 2 (1-\varepsilon)^m$.

Pour k hypothèses, la probabilité que l'une d'elles soit retenue est : $P(h_1 \ldots \vee \ldots h_k) \leq k(1 - \varepsilon)^m$.

Sachant que :

- k est au maximum égal à $|H|$
- pour $0 < \varepsilon < 1$, $(1 - \varepsilon) \leq e^{-\varepsilon}$ et $(1 - \varepsilon)^m \leq e^{-\varepsilon m}$

la probabilité de retenir une hypothèse correcte est : $P(h_1 \ldots \vee \ldots h_k) \leq |H| \; e^{-\varepsilon m} \leq \delta$.

La solution de cette équation par rapport à la taille de l'échantillon m, qui est la complexité de la solution, c'est le nombre d'exemples d'apprentissage qui est nécessaire pour que l'apprenant converge avec une probabilité élevée vers une hypothèse correcte. Cette loi de convergence est le théorème de Blumer [32].

Théorème de Blumer et al. (1987)

Si H est un ensemble d'hypothèses sur un univers X, S un échantillon de m exemples, tirés indépendamment et répondant à une distribution de probabilité D, avec ε et $\delta > 0$, alors si h appartient à H est cohérent avec tous les échantillons d'apprentissage dans S, et

$$m \geq 1/\varepsilon \, [\ln 1/\delta + \ln |H|]$$

Alors la probabilité que h donne une erreur supérieure à ε est inférieure à δ.

3.6. Loi d'Ockham

Dans sa démonstration, Blumer fait intervenir la loi d'Ockham. C'est la loi de parcimonie ou de recherche d'hypothèses de complexité minimum, qui avec une haute probabilité sont prédictives d'observations futures [81].

Lorsque des hypothèses de complexité minimum peuvent être construites dans un temps polynomial de la taille de l'échantillon, on peut construire un algorithme d'apprentissage PAC polynomial.

Si m est polynomial en n : $m = f(n^k)$; $\ln |H| \leq m \quad \Rightarrow \quad |H| \leq 2^{O(n^k)}$.

On obtient une borne supérieure sur l'univers H, pour qu'il existe un algorithme A qui réalise un apprentissage PAC dans un temps polynomial.

Dans un espace d'hypothèses fini, le taux d'erreur est de

$$\varepsilon = \frac{1}{m}\left[\ln|H| + \ln\frac{1}{\delta}\right]$$

Pour aborder des espaces d'hypothèses non-finis, il est nécessaire d'utiliser une mesure autre que la dimension de l'espace d'hypothèses H . Ceci est réalisé par l'utilisation de la dimension de Vapnik et Chervonenkis VC(H) ou $VC_{dim}(H)$.

3.7. La fondation « MACY »

La fondation Macy à New York va organiser entre 1946 et 1953 une série de conférences interdisciplinaires où vont se retrouver notamment Mac Culloch, Pitt, Rosenbluth, Wiener, Shannon, Von Neumann, en compagnie d'autres mathématiciens, psychologues, ingénieurs et neurophysiologistes.

C'est au sein de ce groupe de scientifiques que vont naître et se développer les notions fondamentales de la structure des ordinateurs (Turing, Von Neumann), de la Cybernétique (Wiener), du neurone formel (Mac Culloch) et la théorie de l'information (Shannon).

3.8. Théories de Vapnik et Chervonenkis

Une partition d'un ensemble S est une collection de sous-ensembles disjoints S_i : $\cup$ Si $\equiv$ S.
Une dichotomie d'un ensemble S est une partition de S en deux sous-ensembles disjoints S_1 et S_2 [239] [240].

3.8.1. Shattering

Ce terme désigne l'éclatement par des classes de concepts d'un ensemble d'exemples en toutes ses dichotomies possibles. Nous utiliserons le terme atomiser (ou pulvériser) pour cette fonction.

Soit A une classe d'éléments du domaine D.
Soit B un sous ensemble quelconque de A.
Soit C une classe de concepts.

$$\text{Si} \quad \forall\, B \subseteq A, \quad \exists\, t \subseteq C : t \cap A = B \qquad \text{alors C atomise A.}$$

Pour un nombre d'instances n dans la classe A, le nombre de sous-ensembles de B est 2^n.

Considérons l'exemple suivant :
Soit C l'ensemble des rectangles parallèles aux axes dans R^2.
Soit A = { t,o,m,a } , un ensemble de points non alignés, délimitant un convexe dans R^2.

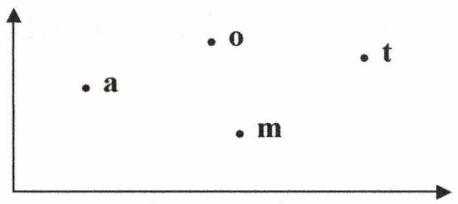


Figure 7 - Shattering

Il existe toujours un rectangle qui englobe n'importe quel sous-ensemble $B_i \subseteq A$.
Pour les 16 sous ensembles $\{B_i\}$ = {{a},{a,t}, {a,m,t}, {}, ... } , il existe un rectangle t tel que $t \cap A = B$ et par conséquent C atomise A.

Dans l'exemple ci-dessus, un groupe de trois points alignés rend impossible l'atomisation, pour un concept rectangle englobant un élément quelconque de l'ensemble. Trois points non alignés peuvent être atomisés en 8 sous-ensembles.

D'autre part, un ensemble de cinq points ne peut plus être atomisé dans R^2.

3.8.2. Dimension de Vapnik-Chervonenkis

La $VC_{dim}(H)$ d'un espace d'hypothèses H, définie sur des instances implicites de l'espace X est la taille du plus grand sous ensemble fini de X qui est atomisé (*shattered*) par H.

Si on se situe dans $\mathbb{R}^2$.

- Pour des rectangles parallèles aux axes, le plus grand sous ensemble de points qui peut être atomisé par ce concept est 4. Donc la dimension VC (H = rectangles) = 4.
- Si on prend comme concept, l'appartenance à un demi-plan, on peut atomiser au maximum 3 points dans un plan, donc VC_{dim} (H = demi plan) = 3
- Si le concept est « être un point» dans un ensemble de points de taille n, la taille de l'espace d'hypothèses est 2^n et la dimension VC_{dim} (H = point) = n.

Théorème de Blumer et al. (1989) : la taille de l'échantillon nécessaire à un système d'apprentissage pour apprendre un concept H, pour un taux d'erreur inférieur à ε avec une probabilité δ est de m exemples :

$$m \geq \frac{1}{\varepsilon}(4\log_2(2/\delta) + 8VC(H)\log_2(13/\varepsilon))$$

C'est la complexité de l'échantillon probablement approximativement correct de l'apprentissage PAC.

Dans ce cas, l'erreur sera :

$$\varepsilon \leq 2\varepsilon_T + \frac{4}{m}\left[d\log_2\left(\frac{2em}{d}\right) + \log_2\left(\frac{4}{\delta}\right)\right]$$

3.8.3. Maîtrise du risque d'erreur

Reprenons le modèle d'apprentissage :

G générateur de vecteurs aléatoires x selon une distribution de probabilité P(x) ;
S superviseur qui génère une sortie y pour chaque vecteur d'entrée x avec une distribution de probabilité P (y|x) fixée, mais inconnue ;
MA une machine apprenante qui implémente des fonctions de type f (x,α) où $\alpha \in \Lambda$, un ensemble de paramètres.

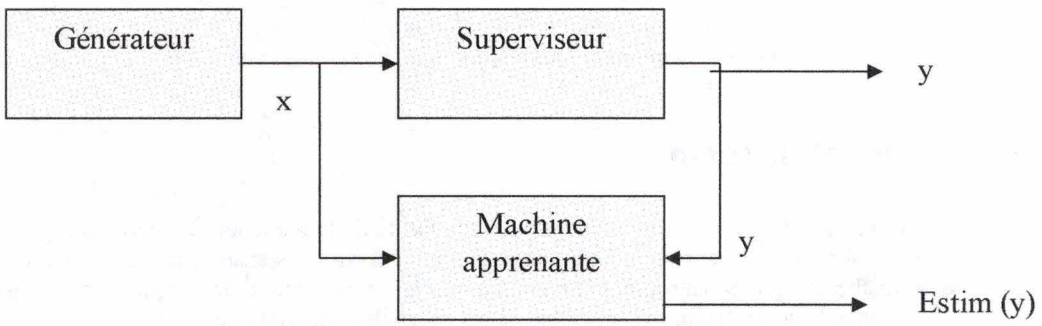


Figure 8 - Modèle d'apprentissage

Minimisation du risque

L'objectif est de trouver une hypothèse $h \in H$ qui minimise le risque réel.
On appelle fonction de coût, la mesure de l'écart entre la sortie réelle y et la valeur estimée de la sortie y.
On considère que l'hypothèse h est liée à la détermination d'un paramètre α et on écrit

$$estim(y) = \hat{y} = h(x) = f(x, \alpha).$$

Soit L[y, f(x, α)] une fonction de coût (loss, perte) et P(x,y) la loi de probabilité jointe sur x et y.
La fonction de coût est $L[y_i,f(x_i, \alpha)] = [f(x_i, \alpha)] - y_i]^2$ dans le cas de la régression.

La valeur totale du risque fonctionnel théorique est :

$$R(\alpha) = \int L[y,f(x, \alpha)]\, dP(x,y)$$

Le but est de trouver α_0 qui minimise le risque fonctionnel R(α).

Ce risque étant impossible à déterminer, car la fonction de probabilité est inconnue, on va minimiser le risque empirique (ERM), qui est le risque observé sur les exemples.

$$ERM = R_{emp}(\alpha) = 1/n \sum_{i=1,n} L[y_i, f(x_i, \alpha)]$$

Cohérence et convergence

Peut-on se baser uniquement sur ERM pour déterminer si un processus d'apprentissage va être cohérent, c'est à dire si on va observer une convergence du risque empirique vers le risque opérationnel réel ? Quelle va être la vitesse de convergence ?

On dit que le processus d'apprentissage est cohérent, si le risque attendu $R(\alpha_l)$ et le risque empirique R_{emp} (α_l) convergent vers la valeur minimale du risque $\inf_{\alpha\in\Lambda} R(\alpha)$.

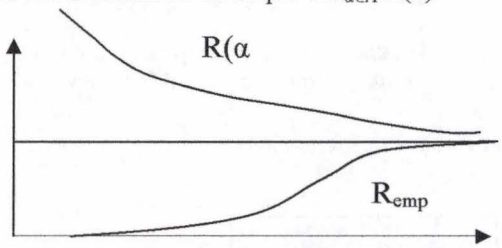


Figure 9 - Convergence des risques

3.9. Entropie et information

Claude Shannon a défini l'information comme une fonction croissante de la réduction d'incertitude qu'elle apporte. Le concept « information » représente une grandeur, sans dimension sémantique, qui s'apparente dans son expression mathématique à l'entropie. C'est Boltzman, qui en mécanique statistique, a établit que l 'entropie est proportionnelle au logarithme du nombre d'états accessibles du système.

$$S = K_B \ln \Omega$$

3.9.1. Entropie de Shannon

Soit la variable aléatoire X d'alphabet $\mathcal{X}$= {x}, distribuée selon p(x) = ℙ [X=x]

L'entropie de Shannon [219] est définie par :

$$H(X) = -\sum_{x\in X} p(x)\log_2 p(x) \quad \text{(en bits)}$$

Etant donné que l'espérance mathématique d'une fonction g(x) est égale à $E(g(x)) = \sum_{x\in X} p(x)g(x)$,

l'entropie peut s'écrire, pour X distribuée selon la loi de probabilité p(x) :

$$H(X) = E[-\log_2 p(x)]_{x\sim p(x)}$$

L'entropie H(X) est une mesure de la quantité d'information, nécessaire en moyenne pour décrire X.

Théorème 1 (Shannon)

Tout signal aléatoire a une complexité irréductible qui implique une limite en dessous de laquelle le signal ne peut être compressé, sans entraîner de perte d'information. C'est l'entropie de Shannon. L'entropie d'une variable aléatoire X est une borne inférieure de la longueur moyenne de la description la plus courte de X, c'est à dire du nombre moyen de questions binaires (oui/non) nécessaires pour identifier complètement X.

$$H(X) \le \overline{l} \le H(X) + 1bit$$

Théorème 2 (Shannon)

Le deuxième théorème définit le codage d'une information de manière à ce qu'un récepteur éloigné puisse décoder cette information « bruitée » avec une probabilité d'erreur nulle. Le message est traduit en mots codes qui doivent être suffisamment éloignés (dans l'espace des séquences possibles) pour que leur version dégradée par du bruit restent distinguables (*'sphere-packing'* dans un espace de grande dimension).

Par ce théorème, Shannon démontre qu'il existe un code qui permet de comprimer l'information de telle manière que la longueur moyenne des mots code L_n soit arbitrairement proche de l'entropie H(x), lorsque $n \rightarrow \infty$.

$$H(x) \leq L_n \leq H(x) + 1/n$$

Défini dans le cadre de la transmission de l'information dans un canal entre un émetteur et un récepteur, le théorème précise aussi que le taux de transfert de l'information doit être inférieur à la capacité du canal.

Axiome de regroupement

Si une variable aléatoire X résulte de choix successifs, alors l'entropie globale est égale à la moyenne pondérée des entropies individuelles.

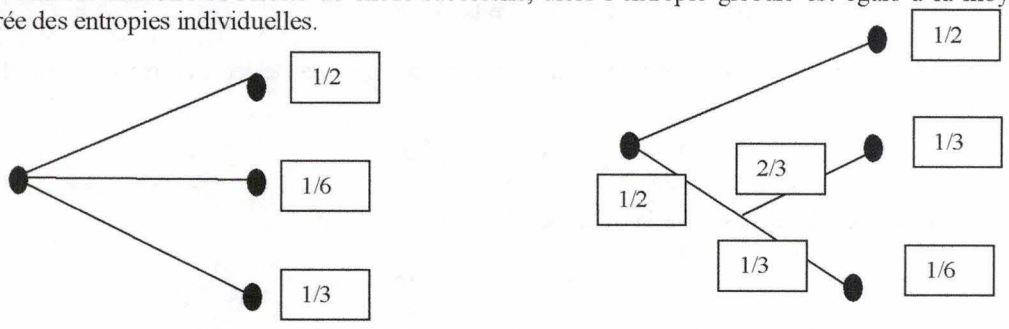


Figure 10 - Entropie globale

$$H(1/2,1/3,1/6) = H(1/2,1/2) + \tfrac{1}{2} H(2/3,1/3)$$

Règle de chaînage

Etant donné deux variables aléatoires, X et Y, non indépendantes l'une de l'autre, on définit l'entropie conditionnelle de Y/X, comme étant l'incertitude sur y, connaissant X

$$H(Y/X) = -\sum_{x \in X} \sum_{y \in Y} p(x,y) \log_2 p(x,y)$$

et la règle de chaînage s'énonce :

$$H(X,Y) = H(X) + H(Y/X)$$

Entropie relative (distance de Kullback)

Soient deux distributions de probabilité p(x) et q(x) définies sur le même domaine X. La distance entre p et q est définie par :

$$D(p//q) \triangleq +\sum_{x\in X} p(x)\log_2 \frac{p(x)}{q(x)} + \sum_{x\in X} E[\log_2 \frac{p(x)}{q(x)}]_{x\sim p(x)}$$

3.9.2. Application à des langages

Dans son article original de 1948, Shannon donne une mesure de l'entropie de la langue anglaise, approximée par un processus stochastique de complexité croissante.

Approximation d'ordre zéro: les symboles (26 lettres+ espace) sont indépendants et équiprobables :

$$H = \log_2 27 = 4{,}76 \text{ bits / lettre}$$

Approximation d'ordre un: les symboles sont indépendants et la fréquence des lettres est celle de l'anglais (exemple: p(E) = 13%, p(Q) = 0,1%). La valeur calculée de l'entropie est :

$$H = 4{,}03 \text{ bits / lettre}$$

Approximation d'ordre deux : on tient compte de la fréquence de paires de lettres (probabilité du doublet « TH » : p(TH) = 3,7%

$$H = 3{,}6 \text{ bits / lettre}$$

Approximation d'ordre trois, basée sur la fréquence de triplets :

$$H = 3{,}2 \text{ bits / lettre}$$

En introduisant des dictionnaires de mots et des probabilités de transitions entre les mots, Shannon établit que le contenu informationnel de l'anglais est de 1,3 bits par lettre.

La compression obtenue ramène donc l'information utile de 4,76 à 1,3 bits par lettre, ce qui correspond à une redondance de 73%. Ceci assure une robustesse importante à la langue écrite, puisque 27% du texte suffit pour reconstituer le contenu du message sans perte d'information.

Analogie avec les protéines

Les protéines sont codées par un alphabet de 20 lettres. Sachant que l'entropie est maximale dans le cas d'une distribution uniforme, on peut calculer une borne maximale pour l'entropie du langage des protéines.

Si on donne à chaque lettre représentant un acide aminé (aa) la même probabilité d'apparition dans une séquence: p(aa) = 0,05

alors la borne maximale de l'entropie est :

$$H(\textit{protéine}) \leq -\sum_{1}^{20} 0{,}05 * \log_2(0{,}05) = \log_2 20 = 4.3 \textit{bits}$$

4. SYSTEMES D'APPRENTISSAGE

4.1. Les Réseaux de Bayes

4.1.1. Bases de la théorie des probabilités

4.1.1.1. *Espace de probabilité*

Soit un ensemble d'échantillonnage Ω contenant n éléments distincts (ensemble fondamental) :
$\Omega = \{ e_1, e_2, \ldots e_n \}$ et une fonction qui assigne un nombre réel P(E) à chaque événement $E \subseteq \Omega$.
P(E) est une fonction de probabilité sur l'ensemble des sous ensembles de Ω [44].

La paire (Ω, P) est un espace de probabilité.

Cet espace dispose des propriétés suivantes :

$P(\Omega) = 1$

$0 \leq P(E) \leq 1 , \forall \ E \subseteq \Omega.$

$\forall \ E, F \subseteq \Omega : E \cap F = \phi \Rightarrow P(E \cup F) = P(E) + P(F)$

Si les événements E et F ne sont pas disjoints : $P(E \cup F) = P(E) + P(F) - P(E \cap F).$

4.1.1.2. *Probabilité conditionnelle et indépendance*

Si E et F sont deux événements tels que $P(F) \neq 0$, la probabilité conditionnelle de E, étant donné F est
$P(E/F) = P(E \cap F) / P(F).$

Si E et F sont indépendants, et sachant que $P(E) \neq 0$ et $P(F) \neq 0$, alors :
$P(E/F) = P(E)$ et $P(F/E) = P(F).$
La condition nécessaire et suffisante est $P(E \cap F) = P(E) \cdot P(F).$

4.1.1.3. *Théorème de Bayes*

On note P (E) et P (F) la probabilité a priori que les événements E et F se produisent. Si on sait que l'événement F s'est produit, la formule de Bayes donne la probabilité a posteriori que l'événement E se produise également, et est notée P(E/F) [70].

Etant donné deux événements E et F tels que $P(E) \neq 0$ et $P(F) \neq 0$, alors

$$P(E/F) = P(F/E) \cdot P(E) / P(F)$$

On peut l'énoncer sous la forme :

$$\text{Probabilité a posteriori} = \frac{\text{Vraisemblance . Probabilité a priori}}{\text{Évidence}}$$

Dans le cas de n événements mutuellement exclusifs et exhaustifs, tels que :

$P(E_i) \neq 0 \ \forall \ i,$

$E_i \cap E_j = \phi \ \forall \ i \neq j$

$E_1 \cup E_2 \cup \ ... \ \cup E_n = \Omega$,

alors $P(F) = \sum^n P(F \cap E_i) = \sum^n P(F/E_i) \cdot P(E_i)$

la formule de Bayes devient

$$P(E_i / F) = \frac{P(F / E_i).P(E_i)}{\sum_{i=1}^{n} P(F / E_i).P(E_i)}$$

4.1.1.4. *Variable aléatoire et distribution de probabilité jointe*

Etant donné un espace de probabilité (Ω, P) , une variable aléatoire est une fonction de Ω qui assigne une valeur unique à chaque élément de l'ensemble fondamental Ω .

$\Omega = \{ e_1, e_2, \ ... \ e_n \}$
$P = \{ P(e_1), P(e_2), \ ... \ P(e_n) \}$
$X = \{ X(e_1), X(e_2), \ ... \ X(e_n) \}$

On note X=x, l'événement { e : X(e) = x } et $P(X=x) = \sum P(e) : X(e) = x$ est la distribution de probabilité de la variable aléatoire X.

Pour deux ou plusieurs variables aléatoires X, Y, ... définies sur le même ensemble fondamental Ω, X=x et Y=y, représentent { e : X(e) = x } ∩ { e : Y(e) = y } et
P (X=x , Y=y) est la distribution de probabilité jointe de A = { X Y } .

4.1.1.5. *Indépendance en probabilité I_p (A,B)*

Si les événements A = a et B = b sont indépendants, on dit A et B sont indépendants en probabilité.

4.1.1.6. *Population infinie et collection d'événements*

L'analyse de données se réalise la plupart du temps sur des ensembles de données de taille importante et pas toujours exhaustive du domaine sous étude. Nous devons donc travailler avec une valeur approchée des probabilités, qui est la fréquence relative définie par Richard von Mises en 1919.

$$\text{probabilité(h)} = \lim_{m\to\infty} \frac{occurrences(h)}{m} = \hat{p}(h)$$

Cette mesure est établie à partir d'un échantillon aléatoire. Elle donne une estimation de la probabilité de l'hypothèse h, telle que : $P(\left|p-\hat{p}\right| \leq \varepsilon) \geq 1-\delta$ pour $m > 2/\delta\,\varepsilon^2$

$\hat{p}$ définit le maximum de vraisemblance de p (ML = *Maximum Likelihood*) . Il s'agit d'une valeur calculée, qu'il ne faut pas confondre avec une probabilité subjective ou un degré de croyance (prévisions) ; La probabilité subjective est aussi appelée probabilité bayésienne, car on fait appel à la théorie de Bayes pour inférer des probabilités inconnues à partir de probabilités connues.

4.1.1.7. *Problème lié à un nombre important de variables*

Pour un ensemble de n variables aléatoires $V = \{ X_1 , X_2 , \ldots X_n \}$, la distribution de probabilité jointe $P = \{ X_1=x_1 , X_2=x_2 , \ldots X_n=x_n \}$, assigne une valeur à toutes les combinaisons des valeurs x_i choisies dans l'espace de X_i.
Lorsque n est élevé, la probabilité conditionnelle d'une variable aléatoire par rapport aux autres n'est pas calculable, car elle fait intervenir un nombre exponentiel de termes 2n-1, et la détermination de la probabilité jointe n'est pas réalisable.
Les réseaux de Bayes vont permettre de représenter la distribution de probabilité jointe d'un grand nombre de variables aléatoires en réalisant des inférences bayésiennes avec ces variables.

4.1.2. Les réseaux bayésiens

Considérons un domaine de connaissances constitué de faits (événements) définis par une série de paramètres, et entre lesquels il existe des relations de causalité. Les paramètres constituent les propriétés de ces faits. Il existe des incertitudes sur la connaissance des propriétés de ces faits, et sur les relations de causalité entre ces événements [187].

Le « bon sens » ou la connaissance intuitive de ces domaines se caractérise par des inférences « floues », et une plus ou moins grande « conviction » de l'exactitude des relations de causalité entre ces événements. Les réseaux bayésiens (*knowledge networks*) permettent la formalisation mathématique (probabiliste) de ce bon sens.

Prenons l'exemple d'un test de diagnostic médical : on veut déterminer le risque qu'un patient présente un cancer du poumon (www.bayesia.com) [150].

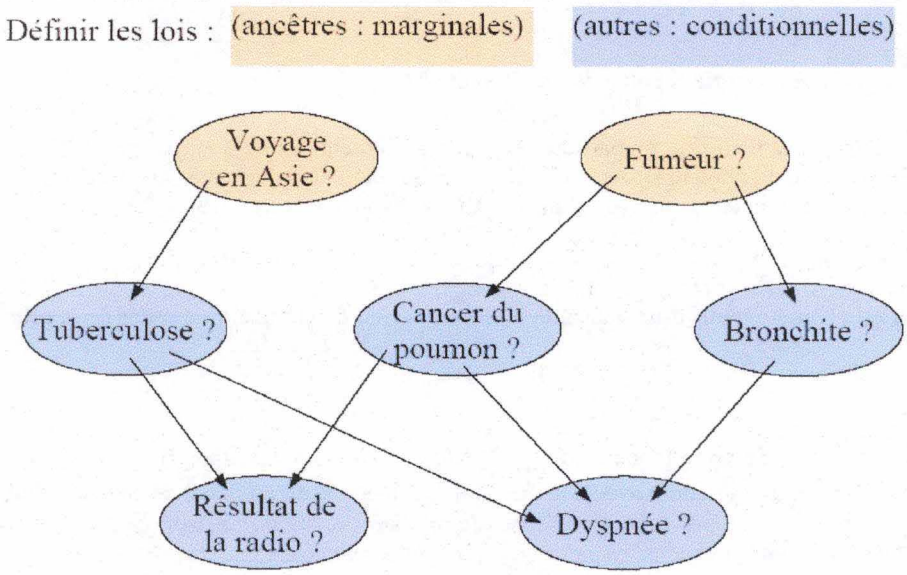
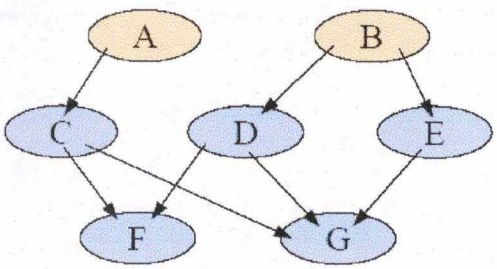


Figure 11 - Réseau bayésien "Assistance en maladies pulmonaires"

On dispose d'une loi de probabilité de l'ensemble des sept variables de notre réseau.



$$[A,B,C,D,E,F,G] = [A][B]$$
$$[C|A][D|B][E|B]$$
$$[F|C,D][G|C,D,E]$$

Figure 12 - Réseau bayésien - loi conjointe

Un réseau bayésien définit la loi conjointe d'un ensemble de variables aléatoires. En général, la loi conjointe est restreinte dans une structure d'indépendance conditionnelle particulière qui est décrite par un graphe acyclique orienté [DAG] construit à l'aide de distributions locales. Aux noeuds ancêtres, on doit fournir la loi marginale ; aux autres noeuds on doit fournir la loi conditionnelle à leurs parents directs. La loi conjointe est le simple produit des lois associées à chaque nœud (théorème de Bayes).

Extrait du site de Netica (http://www.norsys.com)

Figure 13 - Réseau bayésien - connaissance d'expert

4.1.2.1. *Construction et apprentissage des réseaux bayésiens*

Jusqu'à présent nous avons implicitement supposé que nous connaissions ou disposions d'experts pour bâtir un réseau bayésien. L'objectif d'une procédure d'apprentissage est de proposer un (ou quelques) réseau(x) bayésien(s) à partir d'une base de données (en général conséquente).

La construction de familles de réseaux par une association multiple de toutes les variables n'est pas envisageable sur un simple plan combinatoire. Il faut restreindre le problème général en procédant à des simplifications pour faire ressortir les propriétés essentielles.

Les réseaux bayésiens (RB) peuvent être vus comme une généralisation naturelle des processus de Markov (non homogènes) à nombre fini de variables : un RB est une famille de variables aléatoires (souvent discrètes, voire binaires, dans les applications) indexée par un ensemble fini muni d'une structure de graphe sans circuits; la notion markovienne de "dépendance du passé uniquement à travers le passé immédiat" (passé immédiat représenté ici par les prédécesseurs, dans le graphe, de la variable considérée) trouve dans ce cadre une expression naturelle.

4.1.2.2. *Condition markovienne*

Considérons un graphe et définissons N = nœuds, ensemble des sommets du graphe, et A = arcs, l'ensemble des côtés du graphe.

Si P est la distribution de probabilité jointe des variables aléatoires d'un ensemble N, et un graphe acyclique orienté (DAG = « *Directed Acyclic Graph* »), noté $\mathbb{G} = (N , A)$.

On dit que ($\mathbb{G}$, P) satisfait à la condition de Markov, si pour chaque variable $X \in N$, {X} est conditionnellement indépendant de l'ensemble de tous ses non-descendants (ND_X), étant donné l'ensemble de tous ses parents (PA_X).

La condition d'indépendance s'énonce: IP ({X} , ND_X / PA_X).

Si ($\mathbb{G}$, P) satisfait à la condition de Markov, alors P est égal au produit de ses distributions conditionnelles à tous les nœuds, étant donné les valeurs des probabilités de leurs parents.

$$P(X_1, X_2, ...X_N) = \prod P(Xi \,/\, parents(Xi))$$

On peut ainsi établir un parallélisme entre réseaux markoviens et bayésiens

Réseau	markovien	bayésien
Graphe	non orienté	orienté sans circuits
Critère	Séparation : **X** $\perp_s$ **Y** | **Z** si toute chaîne entre **X** et **Y** possède un nœud dans **Z**	d-séparation : **X** $\perp_{ds}$ **Y** | **Z** si $\forall$ chaîne *ch* entre **X** et **Y** $\exists$ un nœud *S* de *ch* t.q. - si *S* est *convergent* sur *ch*. ni *S* ni aucun de ses descendants n'appartiennent à **Z**. - sinon, *S* appartient à **Z**.
Factoris.	si $P > 0$, $P(\mathcal{V}) = \prod_{C \in \mathcal{C}} \psi(\mathbf{C})$, où $\mathcal{C}$ est l'ens. des cliques du graphe.	$P(\mathcal{V}) = \prod_{i=1}^{n} P(X_i \mid Parents(X_i))$

Figure 14 - Parallélisme entre réseau markovien et réseau bayésien

(clique = sous ensemble de nœuds dont toutes les paires sont connectées).

d-séparation

Les nœuds X et Y sont d-séparés par un ensemble de nœuds Z dans G, si Z est intermédiaire entre X et Y, et que Z est un événement observé. Si tous les chemins entre X et Y sont « bloqués » par Z, alors X et Y sont indépendants et d-séparés par Z.

Cette indépendance se note : $X \perp Y \,/\, Z$

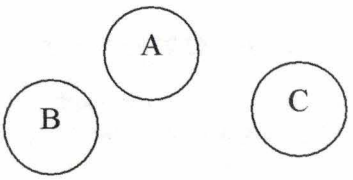
Probabilités aux nœuds du réseau bayésien

Les variables aléatoires associées aux noeuds peuvent être de tout type, en particulier discrètes ou continues. Il peut y avoir différentes manières de représenter une même loi conjointe par un réseau bayésien. La formule de distribution de la probabilité aux nœuds, en application de la loi conjointe et de la d-séparation est :

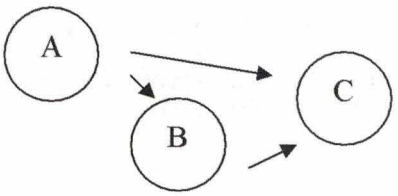
$$P(X_1, X_2, ...X_N) = \prod P(Xi \,/\, parents(Xi))$$

Exemples selon la structure des relations aux nœuds.

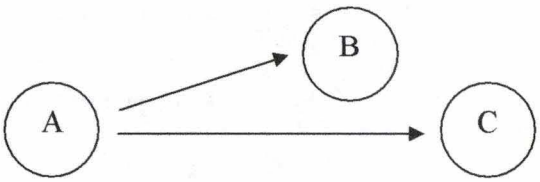
Que des ancêtres : produit des marginales
[A,B,C] = [A].[B].[C]
(A), (B) et (C) sont indépendants.



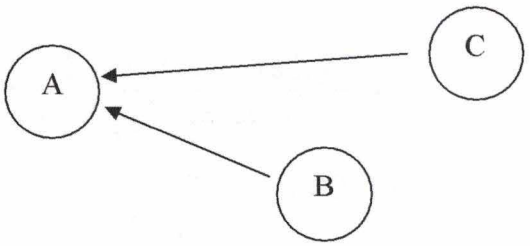
Toutes les flèches (mais pas de cycle !) :
[A,B,C] = [A].[B|A].[C|A,B]
Toute conjointe peut se décomposer ainsi.
Aucune restriction sur la distribution conjointe.
Un réseau de n (n-1)/2 arcs où n est le nombre de noeuds.



Un ancêtre commun, une structure divergente :
[A, B, C] = [A]. [B|A]. [C|A]
(B|A) et (C|A) sont
Conditionnellement indépendants !
Mais (B) et (C) sont «corrélés» !



Un enfant commun, une structure convergente :
[A, B, C] = [A|B, C]. [B]. [C]
alors que (B) et (C) sont indépendants, ils sont conditionnellement dépendants !



Une chaîne :
[A, B, C] = [A]. [B|A]. [C|B]
(A|B) et (C|B) sont indépendants !
Mais (A) et (C) sont «corrélés» !

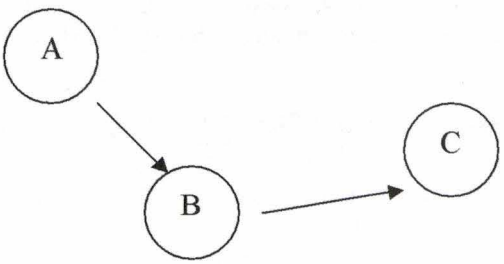


Table 1 - Distribution de probabilité aux noeuds d'un réseau bayésien

Construction automatique et apprentissage du réseau

Cette technique n'est pas développée dans ce travail. On se référera à l'ouvrage de Richard E. Neapolitan, part III Learning (« *Learning Bayesian Networks* ») [96] [187].

4.1.3. Le classifieur « Naïve Bayes »

« Maximum a posteriori hypothesis »

Pour rappel, la formule de Bayes exprime la probabilité a posteriori qu'une hypothèse h soit vérifiée, lorsque on considère les occurrences d'apparition d'une classe de données D.

P(h) = probabilité de l'hypothèse h ∈ H (classe d'hypothèse)
P(D) = probabilité des données D
P(h/D) = probabilité de l'hypothèse, étant donné l'occurrence des données D

On obtient le maximum de la probabilité a posteriori, avec l'hypothèse h_{MAP} qui maximise cette probabilité
P(h/D) = P(D/h) . P(h) / P(D)

$$h_{MAP} = \arg\max_{h \in H} P(h / D)$$

Pour une variable qui est une fonction de plusieurs attributs, la formulation devient complexe et difficile à traiter dans un cas applicatif.

Soit $v \in V$, $v = f(a_1, a_2, \ldots a_n)$, les probabilités sont fonction des corrélations possibles entre les différents attributs :

$$V_{MAP} = \arg\max_{vj \in V} P(v_j / a_1, a_2, \ldots a_n)$$

$$V_{MAP} = \arg\max_{vj \in V} \frac{P(a_1, a_2, \ldots a_n / v_j) \cdot P(v_j)}{P(a_1, a_2, \ldots a_n)} \quad , \quad (\text{avec } P(a_1, a_2, \ldots a_n) = 1)$$

$$V_{MAP} = \arg\max_{vj \in V} P(a_1, a_2, \ldots a_n / v_j) \cdot P(v_j)$$

Si on fait l'hypothèse que tous les attributs sont indépendants, alors le système se simplifie, et on se trouve dans une configuration « naïve bayes » [173].

$$V_{naiveBayes} = \arg\max_{vj \in V} P(vj) \cdot \prod_{i=1}^{n} P(a_i / vj)$$

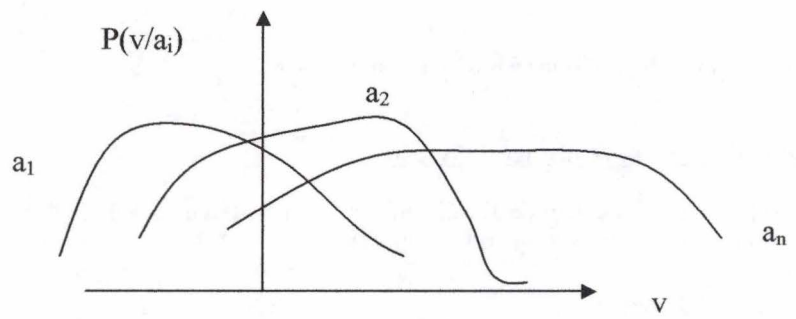


Figure 15 - Distribution de probabilité d'une variable aléatoire

Applications en biologie moléculaire

Outil de classification dans des classes d'appartenance, en fonction d'une liste d'attributs. Il est suffisant d'estimer la distribution individuelle de chacun des attributs Xi pour chaque classe et de les multiplier toutes ensembles.

$$P(X_1, X_2, ..., X_n \mid Class) = P(X_1 \mid Class)P(X_2 \mid Class)...P(X_n \mid Class)$$
$$= \prod_{i=1}^{n} P(X_i \mid Class)$$

Cet outil de classification est caractérisé par deux paramètres, sa sensibilité et sa spécificité. Plus ces deux paramètres auront une valeur élevée, et meilleur sera le modèle.

	PROPOSITION VRAIE	PROPOSITION FAUSSE
PRÉDICTION VRAIE	VP	FP
PRÉDICTION FAUSSE	FN	VN

Table 2 - Sensibilité et spécificité de la loi de Bayes

Sensibilité = VP/(VP+FN)

C'est la fraction de la classe qui est vraie et qui a été correctement prédite par le modèle.
C'est la proportion de vrais positifs que l'on parvient à identifier.

Spécificité = VN/(VN+FP)

C'est la fraction de la classe qui est fausse et qui a été reconnue comme inexacte par le modèle.
C'est la proportion de vrais négatifs que l'on élimine effectivement.

Bayes, Thomas (1763)

An essay towards solving a problem in the doctrine of chances.
Philosophical Transactions of the Royal Society of London,
53:370-418

4.2. Les Modèles de Markov Cachés

4.2.1. Chaîne de Markov

Un processus aléatoire qui est la manifestation d'une suite de changements d'états d'un système et qui ne garde qu'une mémoire limitée du passé est une chaîne de Markov. Une chaîne de Markov est de niveau k, si la probabilité de l'état suivant dépend des k derniers états du système. On peut ramener ces cas complexes à une chaîne de Markov de niveau 1, où la probabilité de l'état suivant ne dépend que de l'état dans lequel se trouve le système.

A une chaîne de Markov est associée une matrice de probabilités de transitions d'un état donné vers les autres états du système. A un instant t, un observateur sait que la chaîne de Markov est dans l'état X_t et il connaît les probabilités d'atteindre l'état X_{t+1} [17] [44] [93].

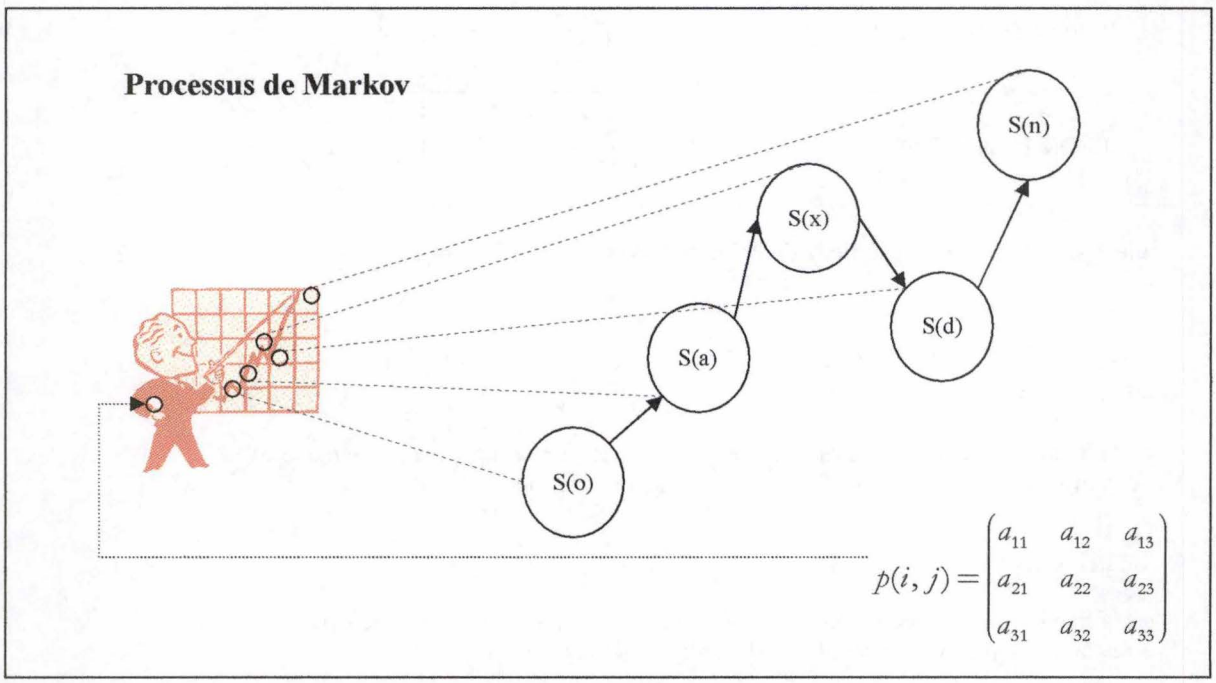


Figure 16 - Processus de Markov

Lorsque le processus aléatoire répond aux caractéristiques d'une chaîne de Markov, mais que l'observateur ne peut connaître les états que prend le système que par des événements indirects, ce processus aléatoire est un processus de Markov Caché (HMM ou *Hidden Markov Model*).

Ce qui est caché pour l'observateur, c'est le modèle lui-même. Il ne peut observer qu'un certain nombre d'événements qui sont la manifestation visible de caractéristiques propres au système. Les paramètres du système sont les probabilités de transition entre états et les probabilités d'émission d'un événement observable.

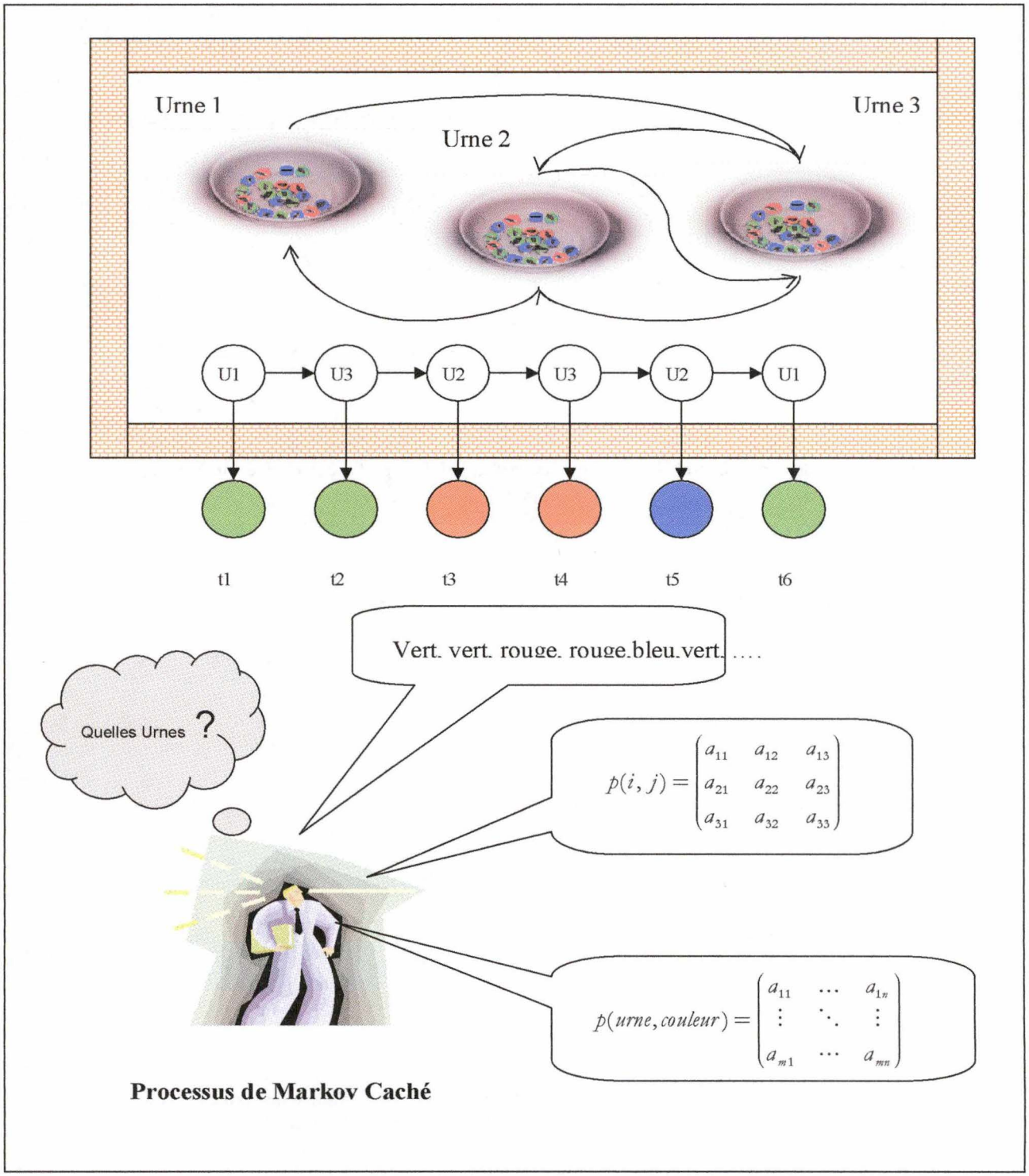


Figure 17 - Processus de Makov caché

4.2.2. Processus de Markov Caché

Processus de Markov

Soit un ensemble d'états $S = \{s_1, s_2, s_3, \ldots s_n\}$ et X_t un processus aléatoire

$\{ X_t \mid t=1,2, \ldots \}$ est une chaîne de Markov, si $\forall$ t , la probabilité de transition de l'état i à l'état j satisfait la propriété :

$p_{ij} = P(X_{t+1} = j \mid X_t = i)$ et p_{ij} indépendant de $X_{t-1}, X_{t-2}, \ldots, X_1$

La matrice de transfert $P(x) = (p_{ij})\ n*n \quad \forall\ i,j \ (1 \leq i,j \leq n)$

répond aux caractéristiques $P = \begin{pmatrix} p_{11} & \cdots & p_{1n} \\ \vdots & \ddots & \vdots \\ p_{n1} & \cdots & p_{nn} \end{pmatrix}$ avec $p_{ij} \geq 0$ et $\sum_{j=1}^{n} p_{ij} = 1$

La distribution initiale définit la probabilité pour que l'état initial de départ d'une chaîne soit l'état i :

$$\pi^0 = (\pi_i^0)_{1*n} \qquad \pi_i^0 = P[X_1 = i]$$

Définition d'un modèle de Markov :

$M = < S, P, \pi^0 >$

Avec :

$S = \{s_1, s_2, s_3, \ldots s_n\}$ un ensemble d'états

$P(x) = (p_{ij})\ n*n$ la matrice de transfert

$\pi^0 = (\pi_i^0)_{1*n}$ la distribution initiale

Processus de Markov Caché

Un processus de Markov Caché est un processus de Markov dont l'observateur ne connaît pas les états successifs, mais est informé d'une série d'observations sur les caractéristiques de ces états.

Soit un processus de Markov $M = < S, P, \pi^0 >$

dont chacun des états Si possède un nombre m de caractéristiques $C = \{ C_1, C_2, C_3, \ldots C_m \}$

La probabilité d'apparition d'une caractéristique C_k, pour k=1,m est spécifique de chaque état S_i, pour i=1,n

La distribution des densités de probabilités des caractéristiques d'un état S_i est

un vecteur $B = (b_i)$ 1*n avec $\sum_{k=1}^{m} b_i(k) = 1$

Ce sont les probabilités d'émission de la caractéristique k pour l'état i.

L'observateur ne connaîtra du système qu'une observation qui lui indiquera quelle est la caractéristique C_k qui caractérise l'état S_i qui est atteint ; Il ne connaîtra pas S_i, il n'observera que C_k.

La séquence d'observation est $O = (o_1, o_2, o_3, o \ldots o_T)$ et les caractéristiques des états satisfont :

$$b_i(k) = P(O_t = C_k \mid X_t = i) \quad \text{et} \quad \sum_{k=1}^{m} b_i(k) = 1$$

Définition d'un modèle de Markov Caché :

$M = < S, O, P, B, \pi^0 >$ de paramètres $\vartheta = < P, B, \pi^0 >$

Avec :

$S = \{s_1, s_2, s_3, \ldots s_n\}$ un ensemble d'états

$O = (o_1, o_2, o_3, o \ldots o_T)$ un ensemble d'observations

$P(x) = (p_{ij})n*n$ la matrice de transfert

$B(x) = (b_{ik})n*m$ la matrice d'émission

$\pi^0 = (\pi_i^{\,0})_{1*n}$ la distribution initiale

4.2.3. Quelles inférences peut-on réaliser avec les MMC ?

Les algorithmes fondamentaux des Modèles de Markov Cachés (MMC) vont permettre de répondre aux questions posées par les problèmes suivants [31] [37]:

Calcul de vraisemblance d'une séquence d'observations :

Etant donné les paramètres ϑ du modèle, quelle est la probabilité de la séquence O d'observations :

$$P(O \mid \vartheta)$$

Identification d'une séquence d'états :

Etant donné les paramètres ϑ du modèle, trouver la séquence d'états S la plus probable générant O :

$$\underset{S}{Arg\max}\, P(S \mid O, \vartheta)$$

Apprentissage supervisé :

Trouver les paramètres qui maximisent la probabilité de génération de ϑ

$$\underset{\vartheta}{Arg\max}\, P(O \mid \vartheta, S)$$

Apprentissage non supervisé :

Trouver les paramètres ϑ qui maximisent la probabilité de génération de la séquence d'observations O dans un modèle donné :

$$\underset{\vartheta}{Arg\max}\, P(O \mid \vartheta)$$

Sans entrer dans de longs développements théoriques, nous allons énoncer les principes des algorithmes qui permettent de répondre à ces questions, et principalement [82] [86]:

- l'algorithme de Viterbi
- l'algorithme de Baum-Welch

4.2.4. Algorithmes fondamentaux

Algorithme Forward-Backward :

Il permet de trouver une solution efficace au calcul de la vraisemblance d'une séquence d'observations. Pour un MMC avec n états et pour une série d'observations O de longueur T, la solution directe de P(O | ϑ) est de complexité exponentielle en T, la longueur de la série d'observations $O\,(Tn^T)$.

Si on pose :

$\alpha_t(i) = P(o_1, o_2, o_3, ..., o_t, X_t = i \mid \vartheta)$

$\beta_t(i) = P(o_{t+1}, o_{t+2}, ..., o_T, X_t = i \mid \vartheta)$

Alors on peut construire un algorithme de complexité $O\,(Tn^2)$, en définissant la probabilité d'observation de la série $O_{|T|} = (o_1, o_2, ..., o_t, o_{t+1}, ..., o_T)$ comme une somme de probabilités d'occurrence de chaque état individuel $X_t = i$ pour la série d'observations $O_{|T|}$ et connaissant ϑ les paramètres du MMC.

Les probabilités peuvent être exprimées par les formules suivantes :

Approche Forward :

$$P(O \mid \vartheta) = \sum_{i=1}^{n} \alpha_T(i)$$

Approche Backward

$$P(O \mid \vartheta) = \sum_{i=1}^{n} \pi_i \beta_1(i)$$

Approche combinée Forward-Backward

$$P(O \mid \vartheta) = \sum_{i=1}^{n} \alpha_t(i)\beta_t(i) \quad \text{avec} \quad 1 \le t \le T$$

Algorithme de Viterbi

Il permet de trouver une solution efficace à la recherche de la séquence d'états la plus probable, étant donné une séquence d'observations $O_{|T|} = (o_1, o_2, ..., o_t, o_{t+1}, ..., o_T)$ et les paramètres ϑ du modèle MMC.

$$\underset{S}{Arg\max}\, P(S \mid O, \vartheta)$$

Il faut trouver la séquence d'états qui maximise la probabilité. On construit un treillis qui représente le déploiement temporel de tous les parcours possibles du Modèle de Markov Caché (chemins de Viterbi).

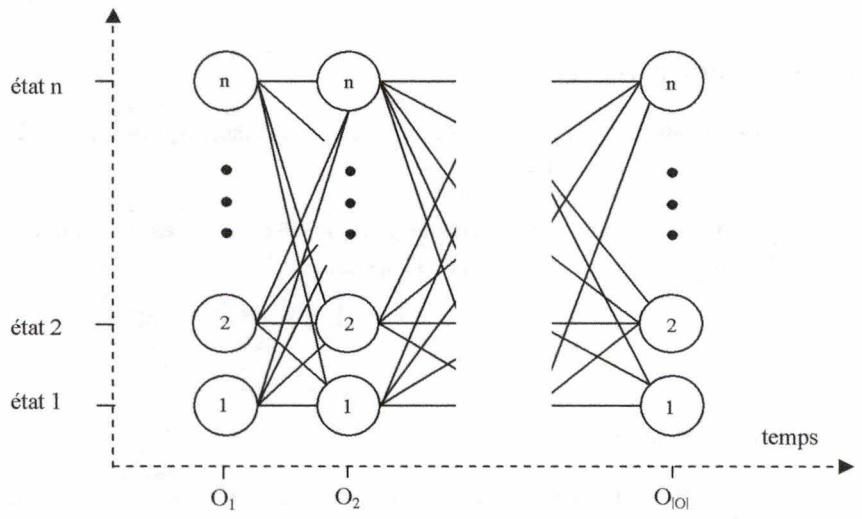


Figure 18 - MMC - Chemins de Viterbi

L'algorithme de Viterbi est un algorithme de programmation dynamique.

Partant de t = 1, pour l'observation O_1, on calcule de proche en proche les transitions pour lesquelles le maximum de la probabilité est atteint.

On procède comme dans l'approche Forward, mais à la place de la somme, on conserve le maximum. On sélectionne l'état S qui maximise $P(O \mid \vartheta)$.

On réalise ensuite une propagation arrière, à partir de l'état $O_{|O|}$, et on sélectionne le meilleur chemin suivant les transitions qui sont marquées.

Estimation des paramètres d'un MMC

Apprentissage supervisé :

Lorsque la séquence d'observations est connue, ainsi que les états correspondants pour une série d'exemples d'apprentissage, tous les paramètres du modèle peuvent être estimés sur base de cet échantillon.

On en détermine une estimation du maximum de vraisemblance pour ϑ : $\hat{\vartheta} = \langle \hat{P}, \hat{B}, \hat{\pi}^0 \rangle$
Sur base des fréquences relatives observées dans l'échantillon.

Apprentissage non supervisé

Lorsque seules les observations sont connues, mais pas les états sous-jacents, l'estimation du maximum de vraisemblance de ϑ est plus complexe à établir.

Pour maximiser $P(O \mid \vartheta)$, on va utiliser l'algorithme de Baum-Welch, qui est une forme de l'algorithme EM (*Expectation-Maximisation*) de maximisation de l'espérance.

Algorithme de Baum-Welch

L'algorithme de Baum-Welch est un algorithme d'estimation itératif. Il maximise la vraisemblance de la séquence d'états selon tous les chemins en relation avec la séquence d'observation $O_{|T|} = (o_1, o_2, ..., o_t, o_{t+1}, ..., o_T)$ qui est générée par le MMC.

Pseudo-compte

Ce sont les variables de comptabilisation, pendant le fonctionnement itératif de l'algorithme.

Positionnons-nous dans le treillis, à l'instant t, pour une transition de l'état S_i vers l'état S_j. La probabilité de transition est :

$$P(X_t = S_i, X_{t+1} = S_j \mid O_T)$$

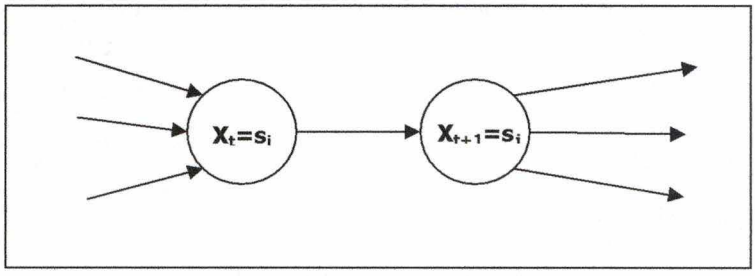


Figure 19 - MMC - Transition entre deux états

Les pseudo-comptes comptabilisent dans l'estimation du modèle, le nombre de fois que la transition $S_i \rightarrow S_j$ apparaît entre l'instant t et l'instant t+1.

Le processus d'itération va déterminer une suite de paramètres $\hat{\vartheta}$ estimés qui va faire croître la probabilité d'observer la séquence d'observation $O_{|T|} = (o_1, o_2, ..., o_t, o_{t+1}, ..., o_T)$

$$P_{\hat{\vartheta}_{t+1}}(O \mid \hat{\vartheta}_{t+1}) > P_{\hat{\vartheta}_t}(O \mid \hat{\vartheta}_t)$$

Soit $s \rightarrow s'$ une transition entre deux états à un quelconque endroit de la séquence d'observation. On va estimer les pseudo-comptes de chaque transition possible, c'est à dire :

$$\hat{P}(s \rightarrow s'), \hat{P}(O \mid s)$$

Compte tenu dans le calcul de :

- l'espérance de passer par s ;
- l'espérance de réaliser la transition $s \rightarrow s'$;
- l'espérance de générer l'observation $O_{|t|}$ dans $s_{(t)}$.

Ces pseudo-comptes, à chaque itération, sont les probabilités de transition données par l'algorithme Forward-Backward.

ETAPES DE L'ALGORITHME

INITIALISATION
EVALUATION
MAXIMISATION
ARRÊT

INITIALISATION

On démarre avec une série de paramètres initiaux $\hat{\vartheta}^0_{t=1}$ qui peuvent être arbitraires.

Soit :

a_{ij} = probabilité de transition $s_i \rightarrow s_j$
$b_j(O_{t+1})$ = probabilité de l'observation O_{t+1} par s_j

EVALUATION

(Pour les observations O_k)

Calcul de la probabilité Forward

$$\alpha_i(t) = P(o_1, ..., o_t, X_t = S_i)$$
$$\alpha_j(t+1) = \sum_{i=1,..,N} \alpha_t(t) a_{ij} b_{j(O_{t+1})}$$

Calcul de la probabilité Backward

$$\beta_i(t) = P(o_{t+1}, ..., o_T, X_t = S_i)$$
$$\beta_i(T) = 1$$
$$\beta_i(t) = \sum_{j=1..N} a_{ij} b_{i(O_t)} \beta_j(t+1)$$

Algorithme Forward-Backward

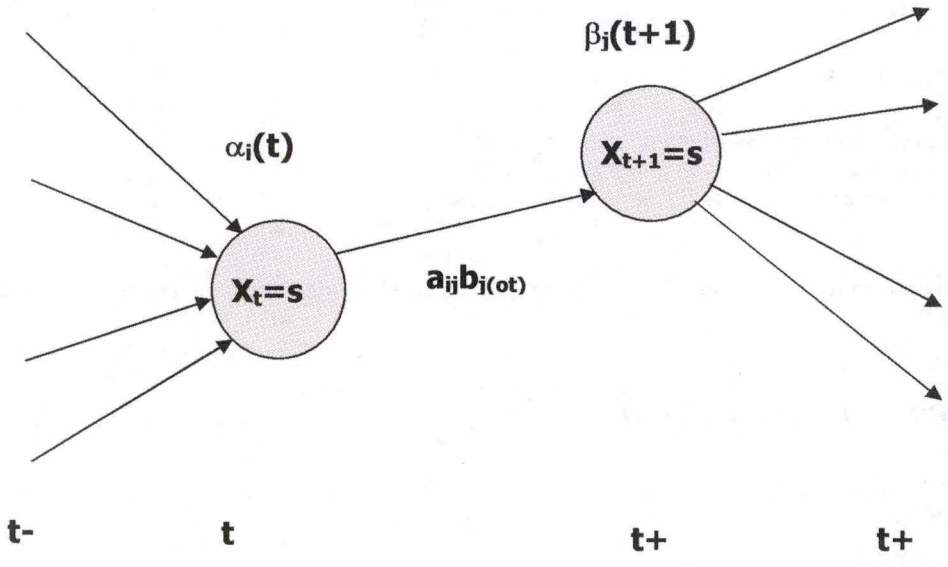


Figure 20 - MMC - Algorithme Forward - Backward

La combinaison de l'approche Forward et Backward donne la valeur du pseudo-compte pour la transition

$$P(X_t = S_i \mid O_{1:T}) = \alpha_i(t)\beta_i(t) / P(O_{t+1:T} \mid O_T)$$

Sommation

Faire la sommation sur T des pseudo-comptes pour déterminer une nouvelle valeur $\hat{\vartheta}$ estimée.

MAXIMISATION

Les estimations de $\hat{P}(s \rightarrow s'), \hat{P}(O \mid s)$ ont permis de calculer une nouvelle valeur $\hat{\vartheta}_{t+1}$ qui va remplacer $\hat{\vartheta}_t$.

On peut démontrer par ailleurs que l'algorithme converge (principe fondamental de l'algorithme EM) et que :

$$P_{\hat{\vartheta}_{t+1}}(O \mid \hat{\vartheta}_{t+1}) > P_{\hat{\vartheta}_t}(O \mid \hat{\vartheta}_t)$$

On boucle entre les phases « Evaluation » et « Maximisation » jusqu'à atteindre le critère d'arrêt.

ARRÊT

Lorsque la probabilité estimée n'augmente plus :

$$P_{\hat{\vartheta}_{t+1}}(O \mid \hat{\vartheta}_{t+1}) - P_{\hat{\vartheta}_t}(O \mid \hat{\vartheta}_t) \leq \varepsilon$$

Ou lorsque on atteint un nombre maximum fixé d'itérations.

MINIMUM LOCAL

Les algorithmes de Viterbi et de Baum-Welch convergent vers un minimum local. L'algorithme de Baum-Welch est cependant plus général que celui de Viterbi, car il explore tous les chemins, alors que Viterbi est basé sur la sélection du meilleur chemin.

4.3. Les réseaux de neurones

4.3.1. Le neurone formel

Issu des travaux de Mc Culloch et Pitt, le neurone artificiel est une image simplifiée d'une cellule du cerveau. Il constitue l'élément de base des réseaux de neurones. On peut le définir comme un petit processeur, un automate à seuil ou une fonction [85].

Par analogie avec le neurone biologique, les connexions en entrée sont appelées les dendrites et la connexion de sortie est l'axone. Les extrémités de contact qui réalisent les échanges d'information sont appelées synapses. Les synapses sur les dendrites, sont les zones d'activation du neurone. Les valeurs d'activation x_i , qui agissent sur les synapses sont pondérées par des coefficients w_i . Dans le neurone, une fonction de transfert S, reçoit la somme Σ de toutes les valeurs d'entrée x_i w_i et produit y, la valeur de sortie.

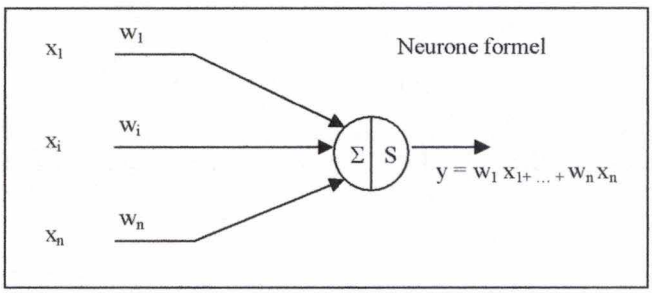


Figure 21 - Neurone formel

Le neurone formel est un automate à seuil, à deux états. La fonction de transfert laisse passer le signal d'activation lorsque celui-ci dépasse un certain seuil, et ne transmet rien sinon.

On définit un neurone comme la représentation d'une fonction y non linéaire, paramétrée, à valeurs bornées, dont les variables sont les valeurs d'entrée (x_i) et les paramètres les poids synaptiques (w_i).

$$y = f(x_1, x_{2,...} x_n ; w_1, w_{2,...} w_m)$$

Le potentiel v du neurone est la combinaison linéaire des variables d'entrée multipliées par les poids, à laquelle se rajoute une constante appelée biais.

$$\nu = \sum_{i=1}^{n} x_i w_i + w_0$$

La sortie du neurone est générée par l'application au potentiel du neurone d'une fonction d'activation.

$$y = f\left[\sum_{i=1}^{n} x_i w_i + w_0\right]$$

Le paramétrage de la fonction peut se réaliser de deux façons : soit par le paramétrage des entrées avec des poids comme dans le neurone formel, soit par des paramètres directement liés à la non linéarité de la fonction d'activation (gaussienne pour réseaux RBF *Radial Basis Fonction,* ou ondelettes).

Caractérisation d'un réseau

Un neurone seul est une simple fonction paramétrique. L'interconnexion de neurones sous forme d'un réseau va permettre de construire des modèles d'environnements complexes. Il va servir d'outil de représentation des connaissances.

Un réseau va être caractérisé par son architecture (types d'interconnexions et fonctions de transfert) et par son mode d'apprentissage.

4.3.2. Types d'architectures

Un réseau est un graphe orienté. Les neurones sont les nœuds et les arêtes sont les connexions entre ceux-ci. Un réseau comporte des neurones d'entrée, des neurones de sortie et des neurones cachés. Il y a échange d'information via les connexions. Le calcul à l'intérieur du réseau est distribué et coopératif.

4.3.2.1. *Réseau de neurones non bouclé*

C'est un réseau où l'information circule de manière unidirectionnelle, des entrées vers les sorties, sans jamais de retour en arrière. La topologie peut être quelconque, cependant la majorité des réseaux sont constitués de couches de neurones successives. On retrouve en littérature anglophone, la dénomination DAG : *Directed Acyclic Graph.*

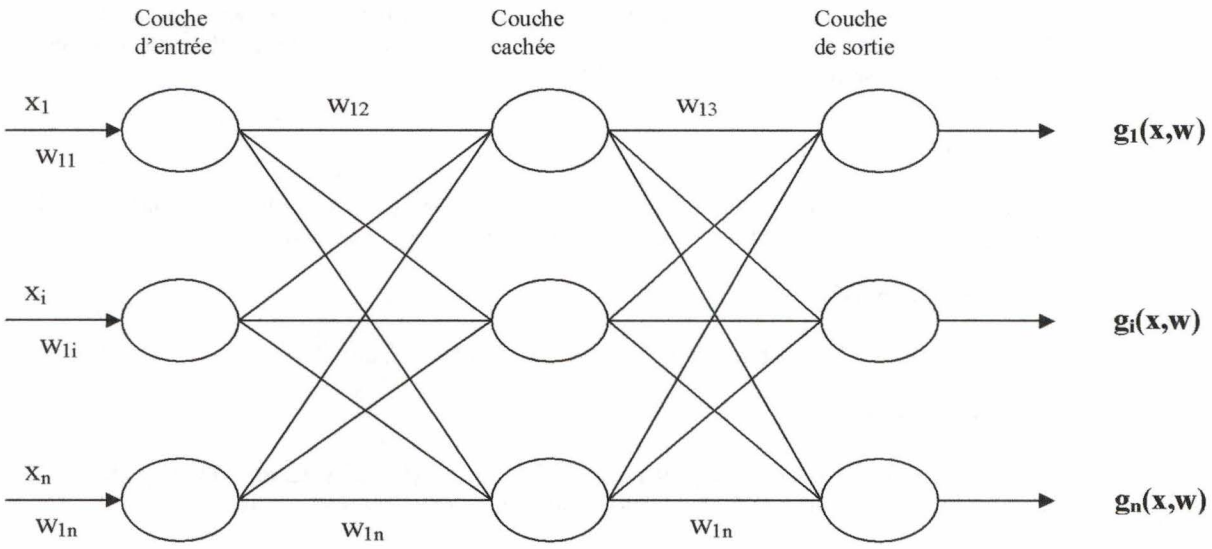


Figure 22 - Graphe acyclique (DAG) du réseau de neurone

Un réseau qui comporte n neurones de sortie réalise n fonctions algébriques des variables d'entrée. Ces fonctions sont le résultat de la composition des fonctions non linéaires réalisées par chacun des neurones.

Un cas particulier des réseaux non bouclés à couches est le perceptron défini par Rosenblatt dont les neurones cachés ont une fonction d'activation sigmoïde (MLP : *Multi Layer Perceptron*).

4.3.2.2. *Réseau de neurones récurrents*

Ce sont des réseaux pour lesquels il existe au moins un cycle dans les connexions. Pour ces réseaux la notion de temps est explicitement exprimée et associée à chaque poids w_i de chaque connexion.

Ces réseaux sont utilisés pour créer des modèles dynamiques de processus, dont les équations qui donnent l'évolution dans le temps sont inconnues ou trop complexes pour effectuer une résolution numérique qui permette de suivre le processus en temps réel. Les équations sont utilisées pour construire le modèle de réseau de neurones, qui va alors pouvoir fournir de bonnes solutions dans des temps de calcul acceptables. On trouve des applications dans le contrôle de fonctionnement de réacteurs ou dans des systèmes d'asservissement.

4.3.2.3. *Réseaux particuliers*

- **Carte auto-organisatrice de Cohonen : le réseau comporte deux couches :**

  La couche d'entrée, ayant un nombre de neurones dimensionné par le système analysé et qui constitue l'espace de représentation.

  La couche de sortie, qui contient un nombre N de neurones et une topologie prédéfinie, correspondant au nombre de classes de généralisation du système étudié.

  La topologie de la seconde couche est déterminée par une fonction de voisinage, qui définit une relation entre chacune des unités et un certain nombre de ses voisines. Cette topologie est constante et n'est pas modifiée par l'apprentissage.

- **Réseau de Hopfield**

  C'est un réseau à une couche, dont chaque nœud est interconnecté à un nombre déterminé d'autres nœuds, par une fonction symétrique qui définit une attraction entre deux nœuds.

- **Réseau de Hamming**

  Calcule une distance de Hamming entre l'entrée et le résultat attendu.

- **Classifieur de Grossberg**

  Carpenter et Grossberg ont développé la méthode ART (*Adaptative Resonance Theory)* pour constituer des classes (*cluster*). Ce sont des réseaux dont l'architecture est évolutive.

La fonction d'activation :

Elle définit la valeur de sortie d'un neurone en fonction des valeurs de ses entrées.

$$v_k >> \varphi(v_k) >> y_k$$

Fonctions linéaires :

Fonction à seuil

$$y_k = \varphi(v_k) = \begin{cases} .1 \text{ si } v_k \geq 0 \\ .0 \text{ si } v_k < 0 \end{cases}$$

Fonction linéaire par parties

$$\varphi(v) = \begin{cases} .1 \text{ si } v \geq \alpha \\ .v \text{ si } \alpha > v > \beta \\ .0 \text{ si } v \leq \beta \end{cases}$$

Fonction sigmoïde :

Logarithmique $\varphi(v) = \dfrac{1}{1 + e^{-av}}$

Tangente hyperbolique $\varphi(v) = \tanh\left(\dfrac{v}{2}\right) = \dfrac{1 - e^{-v}}{1 + e^{-v}}$

Fonctions non linéaires :

$$g(x, w) = \exp\left(\frac{-(x - w)^2}{2\sigma^2}\right)$$

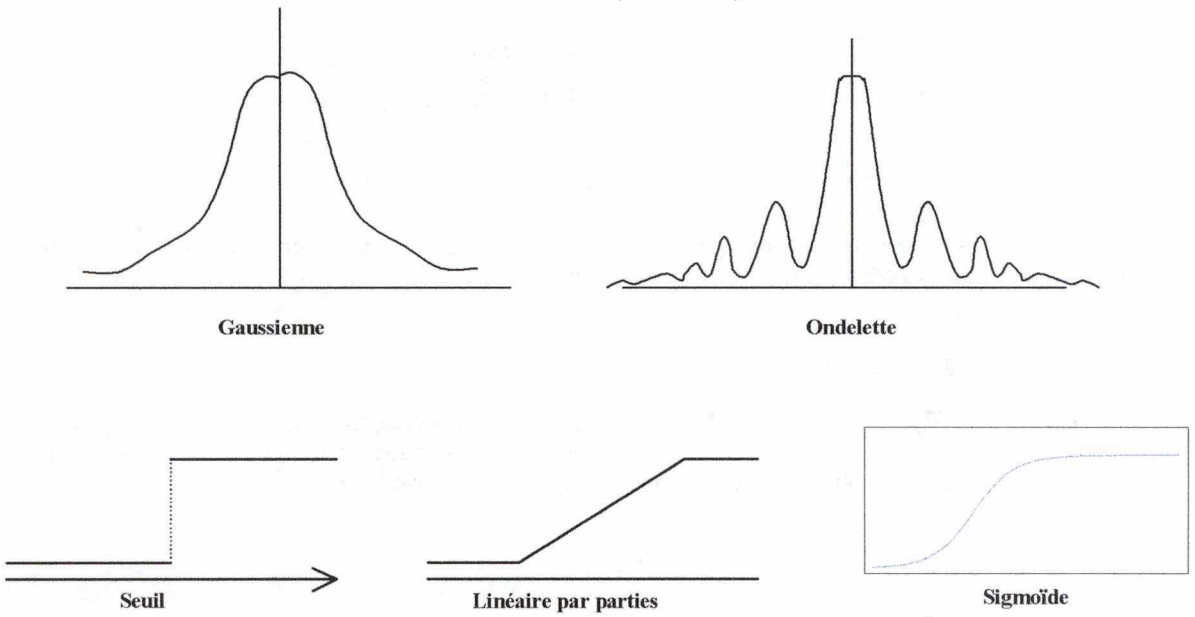


Figure 23 - Diverses fonctions d'activation

4.3.3. Les types d'apprentissage

On précisera la structure du réseau auquel ces différentes méthodes d'apprentissage sont applicables. Les premières méthodes présentées sont historiques.

Loi de Hebb (1949)

C'est un apprentissage non supervisé, par renforcement. Donald Hebb, neuropsychologue, établit la première loi qui va permettre à un réseau de neurones d'apprendre. Il émet l'idée que la modification des forces synaptiques est proportionnelle à l'activation des neurones.

$$\Delta w_{ij} = L\ a_i\ a_j ;\quad L \in [0,1]$$

Cette loi s'applique à des réseaux à deux couches.

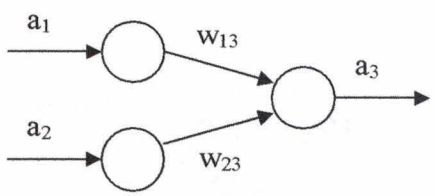


Figure 24 - Loi de Hebb

La variation des poids du réseau, suite à l'apprentissage de l'exemple (a_1, a_2) sera :

$$w_{13} = w_{13} + L\ a_1\ a_3$$
$$w_{23} = w_{23} + L\ a_2\ a_3$$

C'est un apprentissage par corrélation qui ajuste les poids d'une connexion en fonction des valeurs d'activation des neurones connectés et de la vitesse d'apprentissage.

Théorème de convergence du perceptron

Frank Rosenblatt (1957) a étendu l'idée de Hebb à l'apprentissage du perceptron à deux couches ; il énonce une règle d'apprentissage basée sur l'ajustement des poids en proportion de l'erreur entre la valeur du neurone de sortie et la valeur objectif. C'est un mode d'apprentissage supervisé.

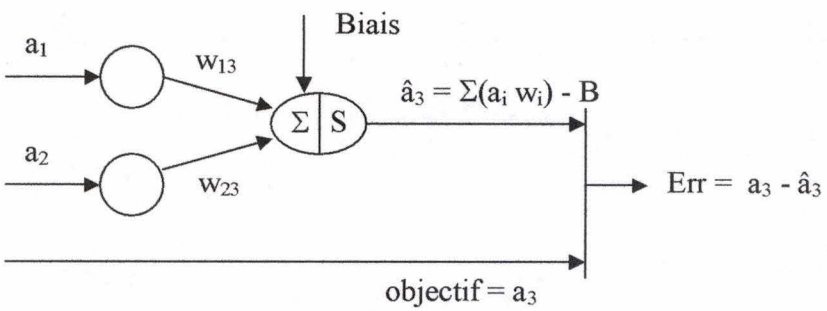


Figure 25 - Convergence du perceptron

Pour un exemple (a_1 a_2) qui devrait donner un résultat correct a_3, l'erreur est l'écart entre le résultat obtenu à travers le perceptron $\hat{a}_3$ et l'objectif à atteindre a_3.

La correction à apporter sur les poids sera proportionnelle à l'erreur et à la valeur d'entrée, modulée par un gain, aussi appelé la taille du pas ou la vitesse d'apprentissage L.

$$\Delta w_i = L * Err * a_i$$

Le théorème de convergence s'énonce:

$$w_{ij} = w_{ij} + L * Err * fc(entrée) \quad \text{avec } 0 < L < 1$$

Cette règle d'apprentissage définit la procédure d'ajustement des poids synaptiques pour réduire l'erreur. On dit que l'apprentissage converge, lorsque l'erreur tend vers une valeur inférieure à un seuil donné.

On démontre que cet apprentissage s'applique lorsque l'on peut faire une partition de l'espace d'hypothèse en deux classes linéairement séparables.

Cependant aucune règle valable qui aurait permis de construire l'apprentissage des poids entre la couche d'entrée et la couche de neurones cachés n'a pu être énoncée à ce moment pour un perceptron à trois couches.

Minsky et Pappert (1969) mettent en évidence les limitations du perceptron à deux couches, sans trouver de moyens de réaliser l'apprentissage du perceptron multicouche.

Apprentissage ADALINE

C'est un mode d'apprentissage supervisé, qui n'est applicable qu'au perceptron à deux couches. ADALINE est la contraction de «*Adaptative Linear Neuron* »

A la différence du perceptron, ADALINE continue à apprendre même si il donne la bonne réponse, car l'apprentissage est basé sur le potentiel d'activation ν, plutôt que sur la valeur de sortie.

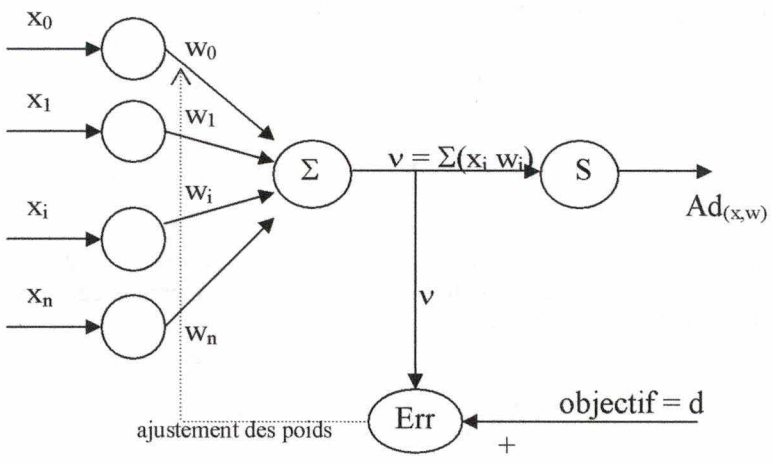


Figure 26 - Adaline

L'erreur considérée est l'écart entre la valeur désirée d et le potentiel d'activation v . La méthode minimise le carré de l'erreur :

$$\min \text{ Err}^2 = \min \ (d-\Sigma \ w_i \ x_i)^2$$

Pour faire tendre le carré de l'erreur vers zéro, on calcule le gradient qui donne la direction de croissance de l'erreur et on adapte les poids en sens opposé.

C'est une méthode de descente du gradient par minimisation du carré de l'erreur.

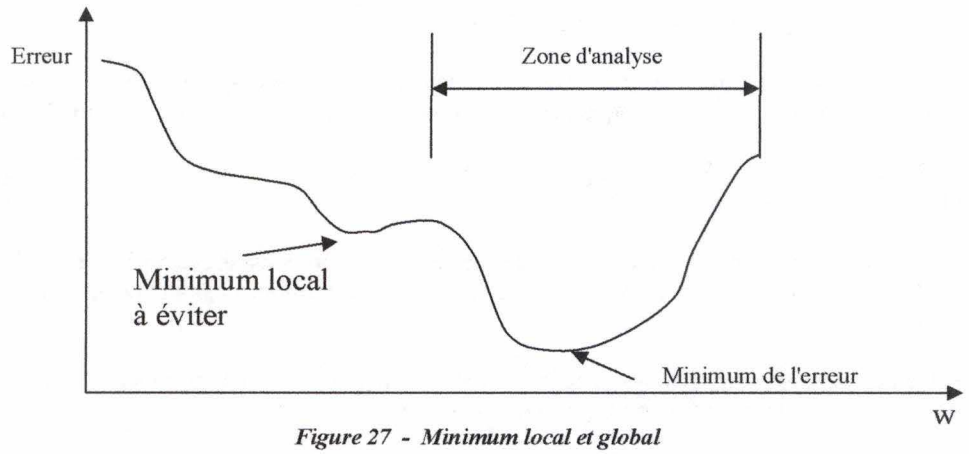


Figure 27 - Minimum local et global

L'erreur est une fonction de w pour une entrée fixée. On calcule les dérivées partielles par rapport à w . Le gradient est donné par :

$$\nabla_w Err^2 = \frac{\partial^2 Err}{\partial w_0^{\ 2}} + \frac{\partial^2 Err}{\partial w_1^{\ 2}} + ... + \frac{\partial^2 Err}{\partial w_n^{\ 2}}$$

La solution de l'équation donne l'ajustement des poids : $(w_i)_{new} = w_i + L \ . \ Err \ . \ x_i$

Sous forme vectorielle, la règle d'apprentissage ADALINE s'énonce:

$$new\_\vec{w} = \vec{w} + L.Err.\vec{x}$$

Règle de Widrow-Hoff

ADALINE est un apprentissage LMS = *Least Mean Square algorithm*. La règle de Widrow-Hoff ou α-LMS améliore l'apprentissage du modèle ADALINE. C'est toujours un apprentissage supervisé, mais on pondère la correction par la taille du vecteur d'entrée :

$$\Delta\vec{w} = L.Err.\frac{\overrightarrow{x}}{\|x\|}$$

On ajuste la valeur des poids en réalisant une classification de la nouvelle valeur x, avec un impact minimum sur les autres entrées.

Règle de Grossberg (1982)

Ce sont des modèles de réseaux de neurones particuliers, caractérisé par :

- Apprentissage en temps réel
- Autoorganisation
- Représentation compacte de phénomènes complexes (encodage Outstar)

Le modèle ART permet de gérer un réseau dont l'évolution dans le temps est significative.

Mémoires associatives de Kohonen (1984)

Kohonen définit un réseau à deux couches, avec comme particularité de réaliser un adressage par le contenu. Dans un tableau a [ ] , le contenu c sert à indexer la mémoire ; l'information c se trouve à l'emplacement a [c] . C'est un mode d'apprentissage non supervisé. Les poids sont ajustés uniquement sur base des valeurs d'entrée.

Extension par Bart Kosko aux mémoires BAM (*Bidirectional Associative Memory* – 1988).
Extension à des réseaux à plusieurs couches de connectivité : le gagnant l'emporte.

Learning Vector quantizer (LVQ)

Apprentissage non supervisé développé par Kohonen. C'est un apprentissage compétitif qui permet aux neurones d'entrée d'activer un seul neurone de sortie.

La classification se fait sur base du plus proche voisin (*k-neighbour*).

La contre-propagation (Hecht-Nielsen -1987)

Méthode basée sur deux méthodes existantes :

- Outstar Encoder de Groosberg
- LVQ de Kohonen

Cette méthode s'applique à un réseau à trois couches. L'apprentissage entre la couche d'entrée et la couche de neurones cachés est réalisée sur base du principe: le gagnant l'emporte.

Réseau de Hopfield (1985)

C'est un réseau à une couche.
Le mécanisme de mémorisation est auto-associatif.

La rétro propagation

Après les premiers travaux sur le perceptron de Rumelhart, Minsky et Pappert, la limitation du perceptron à deux couches à la résolution de problèmes linéaires et suite à l'impossibilité d'entraîner un perceptron avec des couches de neurones cachés, les réseaux de neurones ont été quelque peu négligés.

Enoncée par Werbos en 1974, par Parker en 1982, reprise par Rumelhart en 1986, cette méthode va donner une vie nouvelle aux réseaux de neurones.

La rétro propagation a permis l'apprentissage des réseaux multicouche et la fixation des poids dans les couches cachées.

Quels sont les grands principes de cette méthode ?

- C'est une méthode récurrente, qui réalise un calcul des sorties vers les entrées.
- Les poids de la couche de sortie peuvent être entraînés par l'approche ADALINE.
- C'est une méthode de descente du gradient qui minimise le carré de l'erreur en réalisant un ajustement des poids: $\nabla_w \, Err^2$.
- L'erreur est le résultat de la propagation des erreurs partielles de couches cachées en couches cachées, jusqu'à la sortie du réseau.
- L'erreur dans une couche peut être exprimée en fonction des valeurs d'activation de cette couche (input) et des poids. On démontre qu'une composante du gradient pour un neurone d'une couche cachée, se compose de deux éléments :
  - une variable qui est S, la sensibilité du carré de l 'erreur par rapport au potentiel d'activation en sortie du neurone (à déterminer) ;
  - une variable qui dépend directement des valeurs d'entrée du neurone (connue).

  Tenant compte de ces éléments, on peut alors exprimer la correction des poids à apporter à une couche cachée m, en fonction des valeurs d'activation en entrée du neurone et de la sensibilité S.

$$\Delta w^{m} = - L \, S^{m} \, (a^{m-1})^{T}$$

Partant de la couche de sortie, on va exprimer les dépendances des activations dans la couche m+1 en fonction des activations dans la couche m.

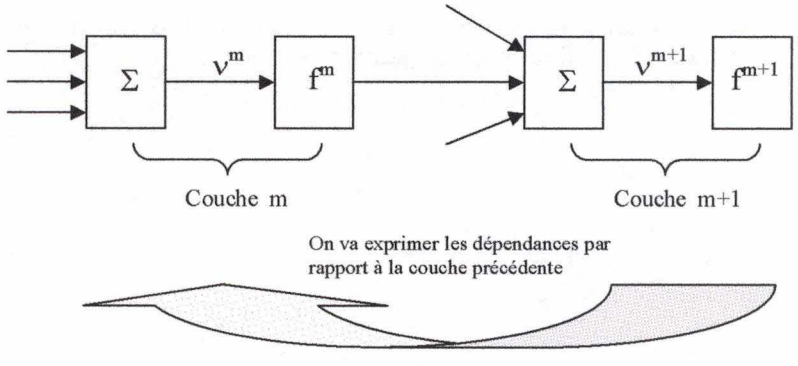


Figure 28 - Dépendances d'activation entre deux couches

Les valeurs des sensibilités de la couche **m** peuvent être exprimées en fonction des sensibilités de la couche **m+1** et de variables connues :

- La dérivée de la fonction d'activation **f'**
- Le potentiel d'activation de la couche m , ν^m
- Les poids de la couche m+1,

La sensibilité s'exprime par :

$$S^m = f'^m(\nu^m) \cdot (w^{m+1})^T \cdot S^{m+1}$$

Partant de la couche de sortie, on effectue le calcul par récurrence, sur toutes les couches cachées, jusqu'à la couche d'entrée.

La rétro propagation est le calcul DUAL de la propagation directe (backward ←-→ forward)

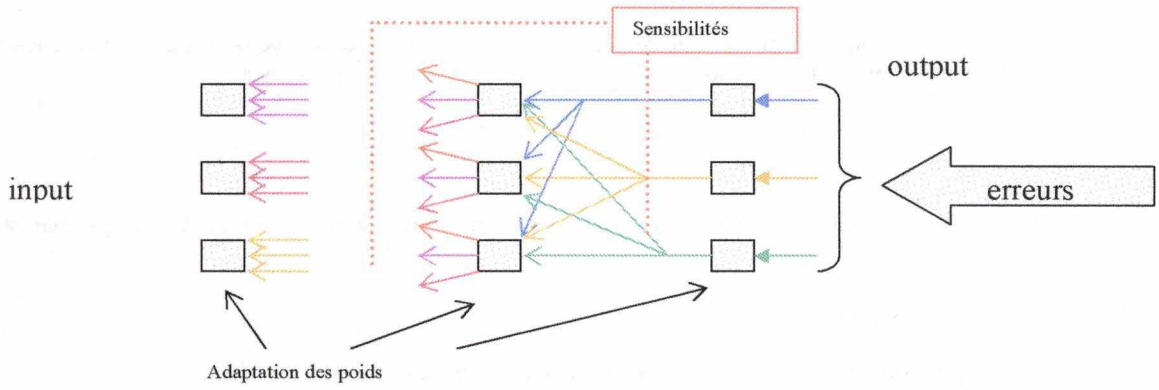


Figure 29 - Rétro propagation

Parallélisme :
Propagation directe : input → output
Propagation arrière : erreur → sensibilité

La complexité et l'exactitude des modèles

Le pas d'apprentissage

La vitesse d'apprentissage doit être modulée en fonction du modèle. Si elle est trop faible, la convergence vers la solution est lente. Si par contre elle est trop grande, il y a un risque d'oscillation autour de la solution.

Les heuristiques de réglage de la vitesse d'apprentissage consistent à :

- diminuer progressivement le pas d'apprentissage, soit manuellement sur base d'observations, soit automatiquement en fonction de la surface d'erreur.

- réaliser une approximation de premier ordre, comme :
  - la rétro propagation avec moment d'inertie
  - delta-bar-delta
  - rprop
- réaliser une approximation du second ordre :
  - la quickprop
  - Leverberg Marquardt

La vitesse d'apprentissage permet de limiter l'impact d'un exemple particulier sur la totalité du modèle. Il est préférable d'ajuster par incréments très petits pour diminuer le risque de détruire une classification qui était correcte pour un nombre élevé d'autres exemples.

Théorème d'approximation universelle

Toute fonction continue sur un domaine fini et borné peut être approximée par un réseau de neurones avec une couche cachée ayant une fonction d'activation sigmoïdale et des sorties linéaires.

$$O(\vec{x}) = \sum_{i=1}^{n_c} \omega_i \sigma(\sum_j \omega_{ij} x_j)$$

Autrement dit, les sommes pondérées des fonctions sigmoïdales des entrées sont des approximateurs universels.

Précision d'approximation

L'erreur d'approximation diminue avec le nombre n_c de neurones cachés : E = Ordre($1/n_c$)

Pour des fonctions d'activation linéaires, et pour une dimension du vecteur d'entrée de d, l'erreur est bornée par : $O(1/n)^{2/d}$
C'est la courbe de dimensionnalité.

Biais inductif

On impose le biais en ajoutant un régulateur à la fonction de coût :

$$E(\vec{w}, \lambda) = \frac{1}{2D} \sum_{\alpha=1}^{D} \sum_{m=1}^{N_0} (t_{\alpha m} - O(\vec{x_\alpha}))^2 + \lambda F(\vec{w})$$

où, $F(\vec{w})$ est le biais choisi et λ est la force d'imposition du biais.

Exemples :

Poids faibles : $F(\vec{w}) = \| w \|^2 = \sum_i w_i^2$

Faible courbure : $F(\vec{w}) = \int \left| \frac{\partial^2 O(x)}{\partial x^2} \right| dx$

Le sous et le surdimensionnement

Erreur de généralisation

Plus le nombre de neurones cachés est grand et plus on entraîne le réseau à rechercher l'optimum local le plus adéquat à une série d'exemples, plus le réseau va être spécialisé à ne reconnaître que des données proches des exemples d'apprentissage.

Si par contre, si il y a trop peu de neurones cachés, il commettra aussi beaucoup d'erreurs car il ne reconnaîtra pas les données, car sa discrimination ne sera pas suffisante.

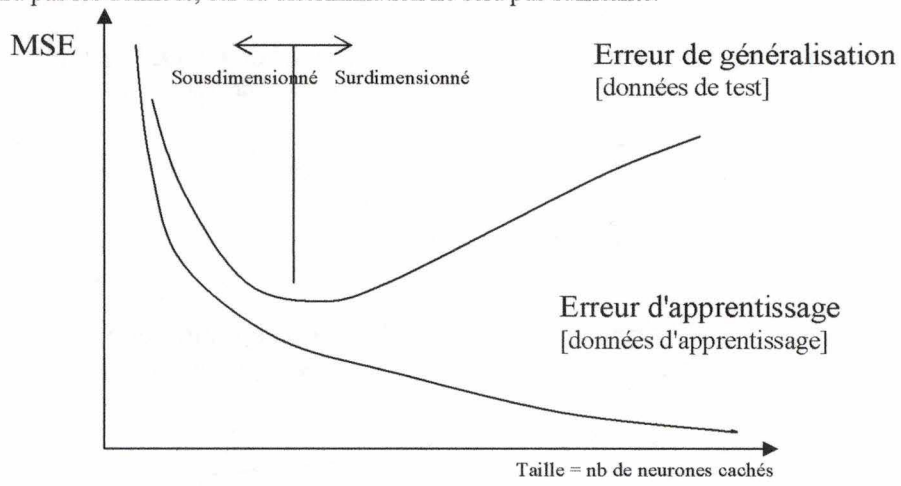


Figure 30 - Erreur de généralisation et taille du réseau

Modèle entraîné jusqu'au minimum local
Modèle de taille variable
Taille de l'échantillon fixe

Influence de la taille de l'échantillon

Lorsque la taille d'échantillon augmente, l'erreur expérimentale tend vers le minimum de l'erreur théorique. Il en est de même pour l'erreur de généralisation. Le graphique ci-dessous donne l'évolution des erreurs pour un modèle entraîné jusqu'au minimum local.

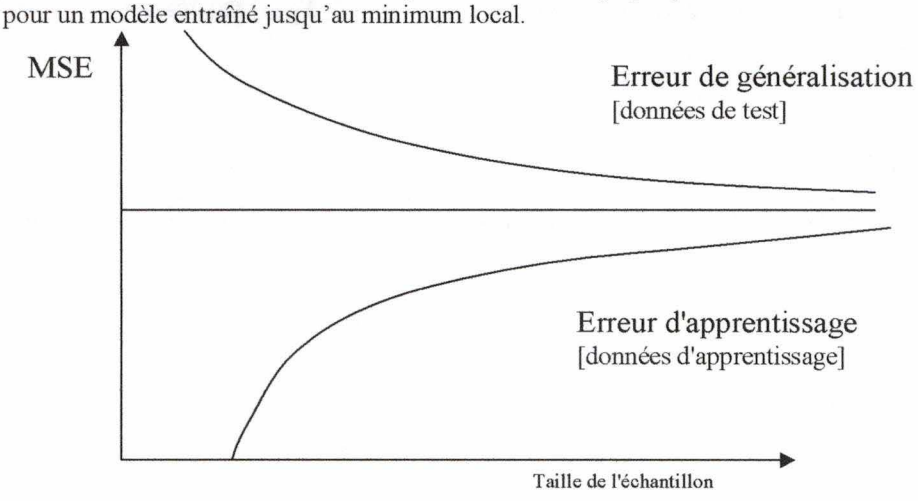


Figure 31 - Influence de la taille de l'échantillon

Modèle de taille fixe
Taille de l'échantillon variable

Nombre d'itérations

On retrouve une erreur de généralisation du même type que celle liée au dimensionnement de la couche cachée [53].

L'allure des courbes d'erreur est donnée par le graphique ci-dessous.

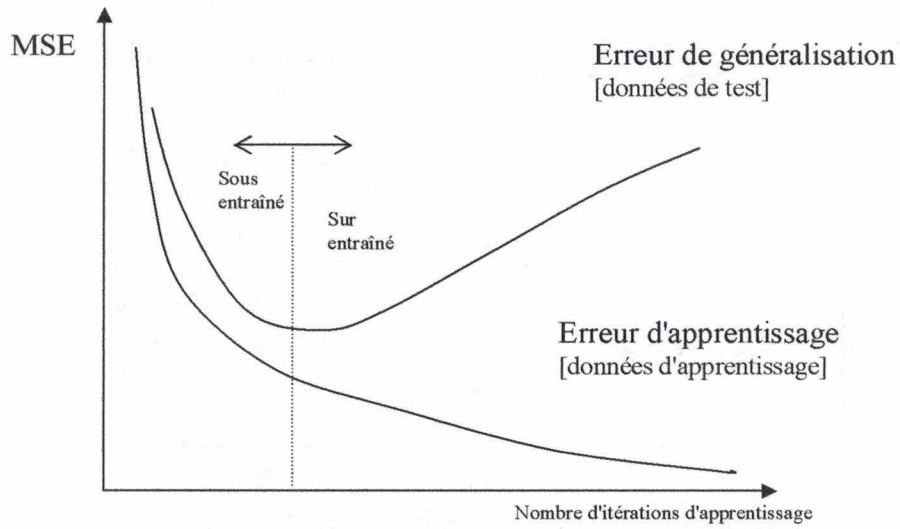


Figure 32 - Erreur de généralisation et nombre d'itérations

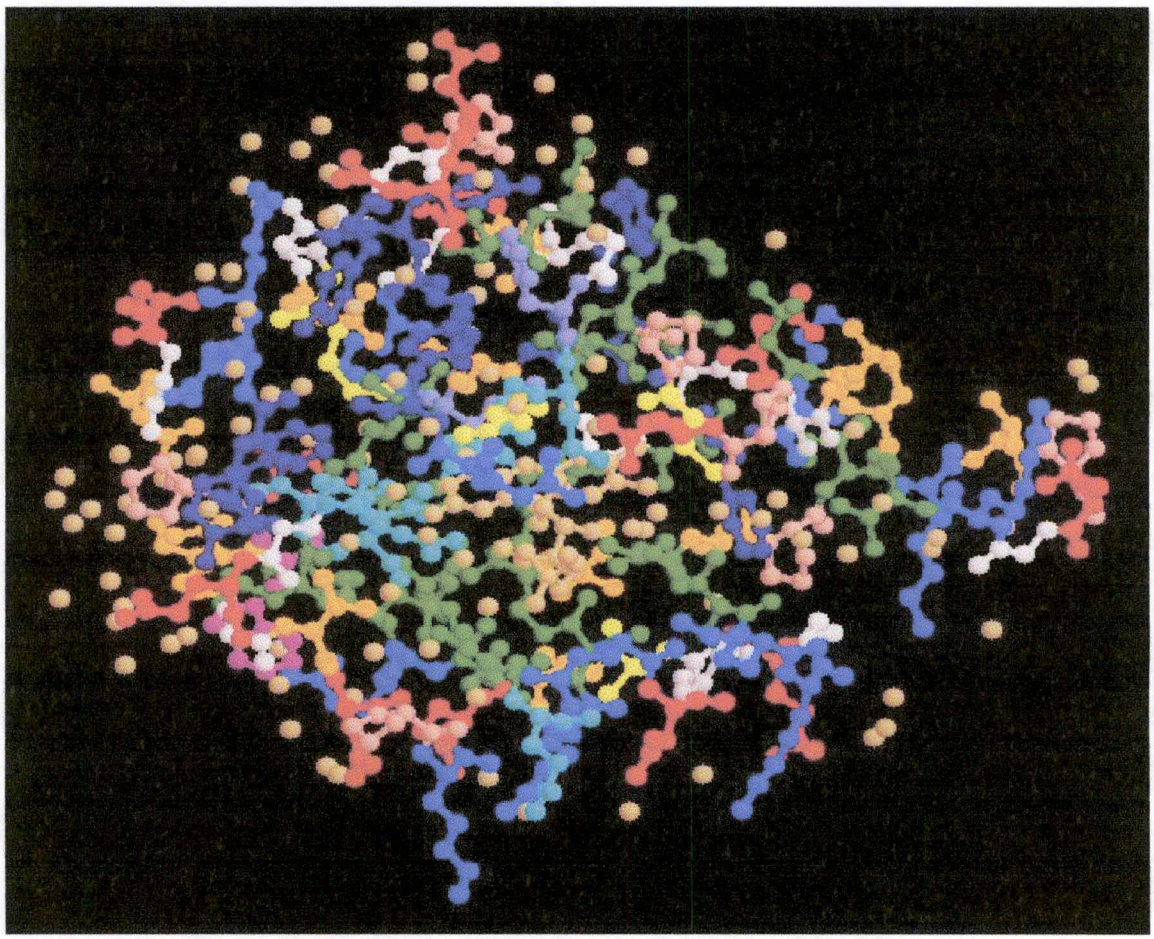
Modèle de taille fixe
Taille de l'échantillon fixe

Le nombre d'itérations est croissant, et les échantillons sont constitués à partir du même set d'apprentissage.

Théorie statistique

L'application de la théorie de l'apprentissage statistique aux réseaux de neurones est développée à l'Annexe A.

APPLICATION EN INGENIERIE DES PROTEINES



5. LES PROTEINES

5.1. L'information génétique

Les chromosomes

Les chromosomes sont constitués de deux brins d'acide désoxyribonucléique (ADN) enroulés en spirale, fortement repliés sur eux-mêmes et que l'on trouve dans les cellules de tout organisme vivant. Ce sont des chaînes de nucléotides extrêmement longues, dont moins de dix pourcents contiennent l'information génétique, codée dans des séquences que l'on appelle les gènes [4] [27].

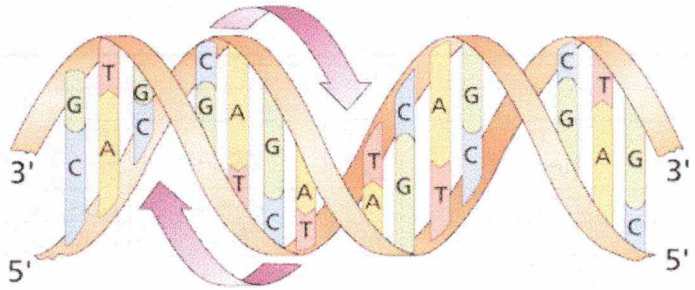


Figure 33 - ADN et chromosomes

L'ADN est un assemblage de quatre monomères, les nucléotides, qui portent des bases différentes: l'adénine (A), la thymine (T), la cytosine (C) et la guanine (G). Les bases sont toujours en correspondance deux à deux, dans les deux brins du chromosome. Ceux-ci sont donc l'image miroir l'un de l'autre. On parle de paires de bases, et le génome humain en compte plus de trois milliards.

Les gènes

Les gènes sont des morceaux de séquence d'ADN, d'une longueur variant de moins de mille paires de bases jusqu'à près de un million de paires de bases, situés à des endroits particuliers des chromosomes. Ils contiennent l'information nécessaire à la cellule pour fabriquer d'autres molécules, les protéines.

Sur réception de signaux particuliers de la cellule, des empreintes de gènes sont créées, ce sont les ARN_m (Acide RiboNucléique- ARN messager), qui pour les cellules eucaryotes vont passer du noyau dans le cytoplasme. L'ARN_m va s'intégrer à une unité de fabrication, le ribosome qui va produire la molécule de protéine pour laquelle l'ARN_m est programmé. Lors de la transcription de l'ADN en ARN_m la base T (Tyrosine) est remplacée par une autre base, l'Uracile (U).

Une protéine est le résultat de la traduction de l'information génétique contenue dans un gène.

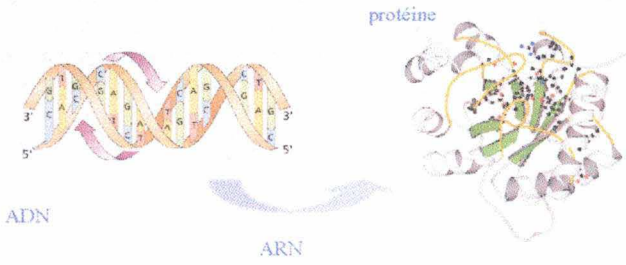


Figure 34 - Gènes et protéines

Le code génétique

Les mots du langage de transcription de l'information contenue dans l'ADN sont les codons. Un codon est un groupe de trois bases consécutives, ce qui permet de réaliser 64 permutations (4x4x4). Il y a cependant une forte redondance au niveau des codons et le code de sortie ne correspond plus qu'à vingt molécules (acides aminés) qui associées en chaînes constituent les protéines [99] [200].

La table des correspondances entre codons et protéines est un code de transcription qui est quasiment universel pour tout le monde vivant. Cette table est le code génétique : lorsque l'on connaît la position d'un codon dans un gène, on connaît alors aussi la position de l'acide aminé correspondant dans la protéine codée par ce gène.

Le code génétique (pour les eucaryotes) est donné dans la table ci-dessous. On notera qu'il existe aussi des codons de pilotage de la transcription (stop).

PREMIERE LETTRE	DEUXIEME LETTRE U		DEUXIEME LETTRE C		DEUXIEME LETTRE A		DEUXIEME LETTRE G		TROISIEME LETTRE
U	UUU	phénilalanine	UCU	sérine	UAU	tyrosine	UGU	cystéine	U
	UUC		UCC		UAC		UGC		C
	UUA	leucine	UCA		UAA	codons stop	UGA	codon stop	A
	UUG		UCG		UAG		UGG	tryptophane	G
C	CUU	leucine	CGU	proline	CAU	histidine	CGU	arginine	U
	CUC		CGC		CAC		CGC		C
	CUA		CGA		CAA	glutamine	CGA		A
	CUG		CGG		CAG		CGG		G
A	AUU	isoleucine	ACU	thréonine	AAU	asparagine	AGU	sérine	U
	AUC		ACC		AAC		AGC		C
	AUA		ACA		AAA	lysine	AGA	arginine	A
	AUG	méthionine	ACG		AAG		AGG		G
G	GUU	valine	GCU	alanine	GAU	acide aspartique	GGU	glycine	U
	GUC		GCC		GAC		GGC		C
	GUA		GCA		GAA	acide glutamique	GGA		A
	GUG		GCG		GAG		GGG		G

Figure 35 - Code génétique

La correspondance entre codons et protéine est inscrite dans des molécules stables qui sont les ARN de transfert. L'ARN_t comporte deux sites réactifs : un site de reconnaissance du codon de l'ARN_m et un site de capture d'un acide aminé spécifique.

Lors de la synthèse d'une protéine, l'ARN_m entre en association avec un ribosome. Sur le premier codon de l'ARN_m, vient se placer le codon miroir de l'ARN_t, qui porte son acide aminé spécifique et ainsi de suite sur les codons suivants. L'acide aminé de chaque ARN_t se lie immédiatement à celui qui le précédait dans la chaîne de la protéine. L'ARN_t, qui a rempli sa fonction repart et l'ARN_m avance d'un codon dans le ribosome [35].

Illustrons ce mécanisme sur le schéma suivant :

Prenons comme exemple le premier acide aminé (AA) de la protéine, la méthionine (Met). Cet AA est identifié par le codon ATG sur l'ADN dans le gène, et sa transcription dans l'ARN_m est AUG, qui est son code génétique que l'on retrouve dans le tableau de la page précédente.

Lors de la traduction de l'ARN_m dans le ribosome, on va construire une protéine avec la méthionine comme premier acide aminé. Le deuxième acide aminé est l'arginine et le troisième est l'alanine.

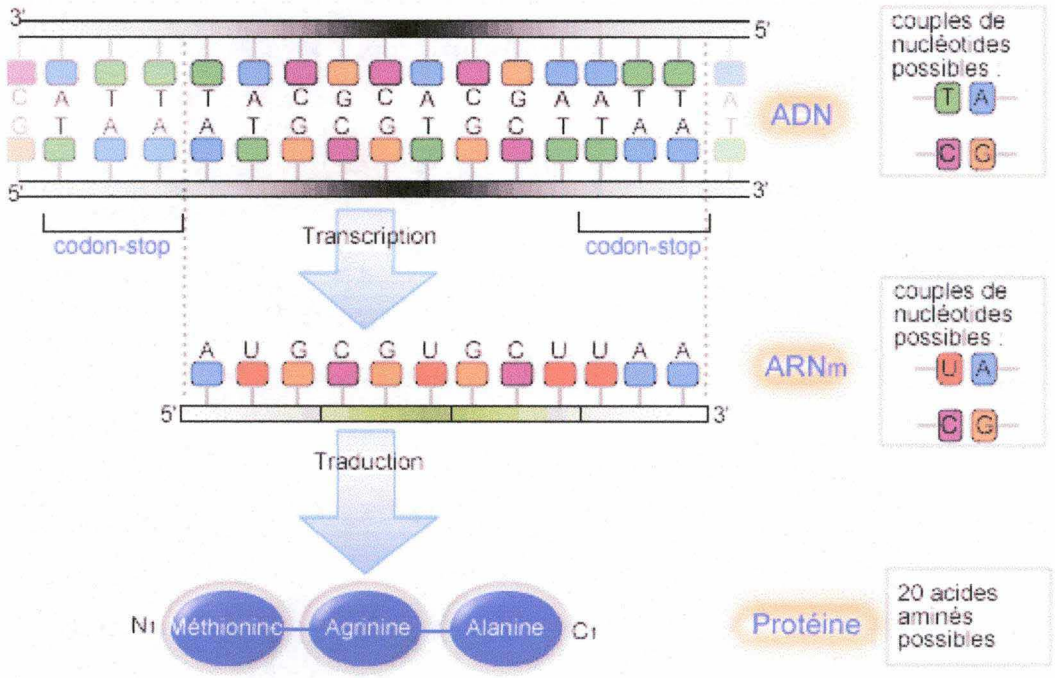


Figure 36 - Transcription et traduction

Source : http://fr.wikipedia.org/wiki/Image:Prot%C3%A9ines.png

Voyons en détail le mécanisme de traduction au niveau du ribosome dans le second schéma, ci-après.

Le ribosome est représenté par ses deux sous unités colorées en jaune et en vert. Sur la chaîne **5' – AUG GAA GCC AGC CUG CCA UGG GAA – 3'** de l' ARN_m , vient se placer sur la première position **AUG**, l'ARN_t de la méthionine avec son codon miroir **UAC**, suivi d'un deuxième ARN_t qui code pour l'acide glutamique, d'un troisième ARN_t qui code pour l'alanine. Ces acides aminés s'associent en une chaîne peptidique qui est la traduction du gène de la protéine.

Le mécanisme moléculaire fait intervenir des séquences d'amorçage **NNNNN** et des molécules annexes (IF3, RF), utiles pour le pilotage de ce mécanisme.

Etapes cytoplasmiques de traduction de ARN_m en protéine

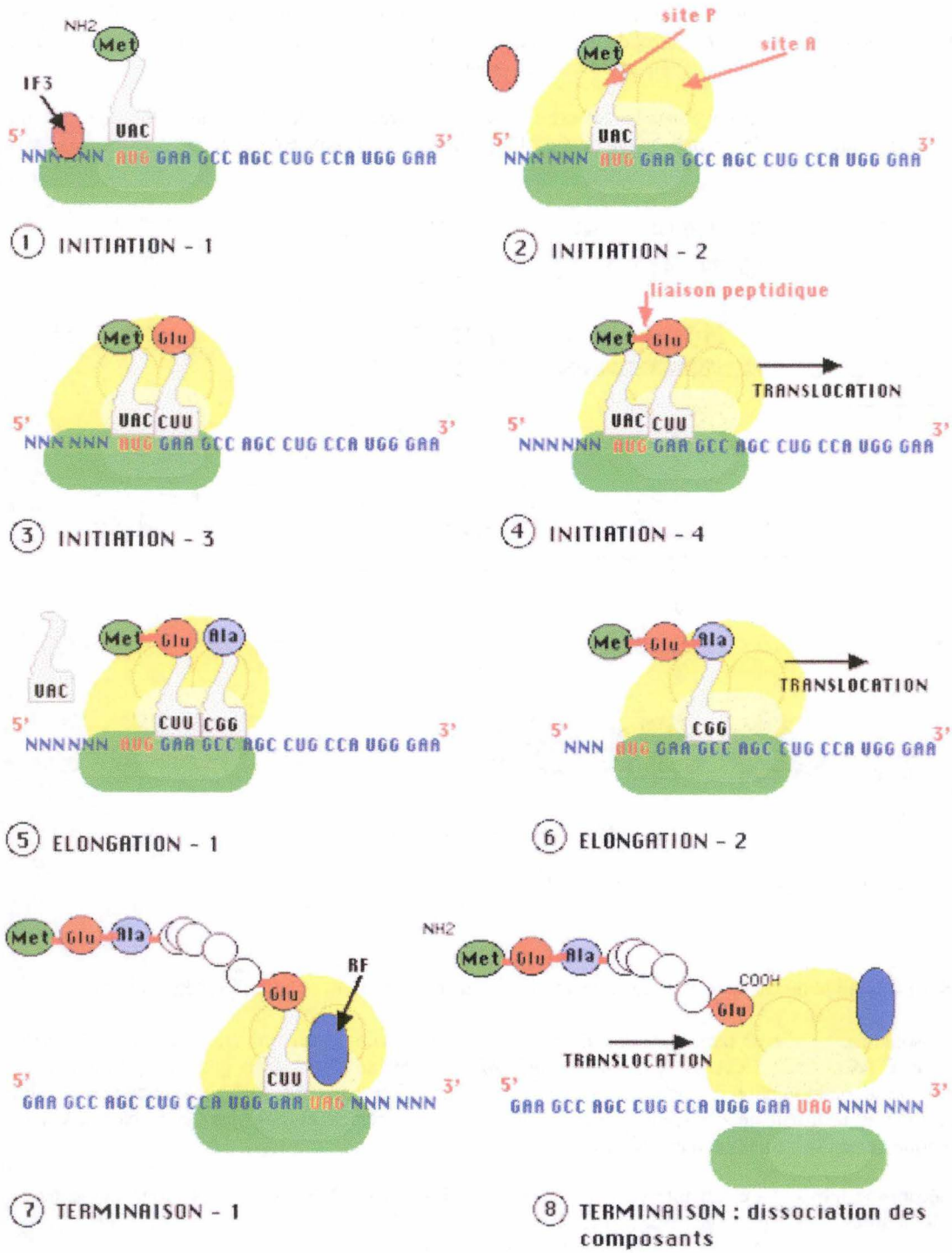


Figure 37 - Ribosomes et protéines

source : http://ead.univ-angers.fr/~jalouzot/genetique/courshtm/chap5/chap5-3.htm

5.2. La structure des protéines

protéine, de πρωτειοσ (au premier rang)

Les acides aminés

Les acides aminés sont représentés par une lettre (voir tableau ci-dessous); ils sont caractérisés par leur masse, leur volume, leur charge électrostatique et leur taux de présence dans les protéines.

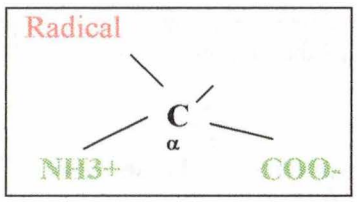
Residu	Masse (Dalton)	Volume de van der Waals A°(^{3})	Présence dans les protéines (%)
Ala (A)	71.09	67	8.3
Arg(R)	156.19	148	5.7
Asn(N)	114.1	96	4.4
Asp(D)	115.09	91	5.3
Cys(C)	103.15	86	1.7
Gln(Q)	128.14	114	4.0
Glu(E)	129.12	109	6.2
Gly(G)	57.05	48	7.2
His(H)	137.14	117	2.2
Ile(I)	113.16	124	5.2
Leu(L)	113.16	124	9.0
Lys(K)	128.17	135	5.7
Met(M)	131.19	124	2.4
Phe(F)	147.18	135	3.9
Pro(P)	97.12	90	5.1
Ser(S)	87.08	73	6.9
Thr(T)	101.11	93	5.8
Trp(W)	186.21	163	1.3
Tyr(Y)	163.18	141	3.2
Val(V)	99.14	105	6.6
Moyenne pondérée	119.4	161	

Figure 38 - Les acides aminés

Source : http://www.biochimie.univ-montp2.fr/licence/qabs/peptides/ac_amine.htm

Les vingt acides aminés de base sont construits autour d'un même modèle générique [206]. Ils sont composés de :

- Une zone identique: un carbone central appelé C_α sur lequel sont fixés un groupement carboxylique -COOH et un groupement amine -NH2. Ces deux groupements sont réactifs. Ils vont réaliser des liens peptidiques (-CO-NH-) entre les C_α des acides aminés et constituer l'ossature de la protéine.
- Une zone constituée d'un radical (R_i) qui est variable d'un acide aminé à l'autre et qui les différentie.



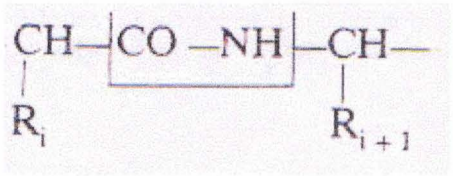


Figure 39 - Lien peptidique

La structure primaire

La protéine, lorsqu'elle est libérée par le ribosome dans le cytoplasme est une chaîne d'acides aminés, dont les C_α sont reliés par un lien peptidique. La séquence de ces acides aminés est appelée structure primaire de la protéine. Elle ne tient compte que de l'ordre des acides aminés dans la protéine.

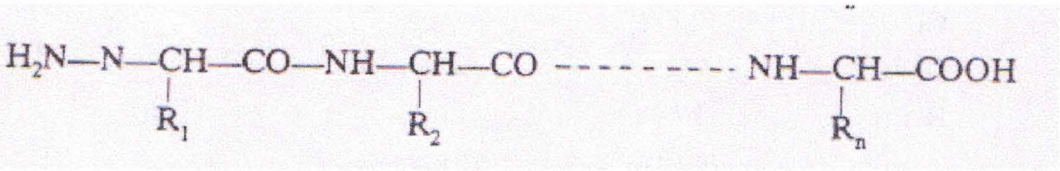
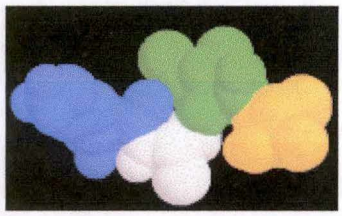


Figure 40 - Structure primaire

On remarquera que si les extrémités terminales sont différentes, l'ossature constituée par les C_α et les liaisons peptidiques est identique sur toute la longueur de la protéine.

La structure primaire d'une protéine est généralement représentée par une séquence de lettres identifiant chaque acide aminé. Par exemple un peptide constitué de quatre acides aminés : lysine – alanine – isoleucine – tyrosine est représenté par une chaîne de quatre lettres : **KAIT,** alors que sa structure réelle dans l'espace est bien plus complexe.



Structure du peptide KAIT, représentée à l'aide du logiciel « CHIME »

La structure secondaire

La chaîne protéique se replie sur elle-même et forme des configurations particulières (hélices, tournants, plans) qui constituent ce qui est appelé sa structure secondaire. La chaîne de la structure primaire est soumise à des contraintes internes et externes, qui vont agir sur sa conformation. Ce sont des forces qui par un jeu d'attractions et de répulsions vont conduire la molécule vers un potentiel minimum. On distingue quatre types de forces:

- L'effet hydrophobe: en solution aqueuse, les zones sans aucune charge électrostatique ne peuvent se mélanger à l'eau; elles ont tendance à s'agglomérer.
- Les liaisons ioniques: les zones qui sont chargées électriquement (ions) ont leurs charges opposées qui s'attirent.
- Les liaisons hydrogène: force d'attraction entre un hydrogène et un oxygène d'un autre acide aminé de la chaîne.
- Les ponts disulfure: les acides aminés qui possèdent un atome de soufre, peuvent former une liaison covalente (-S-S-).

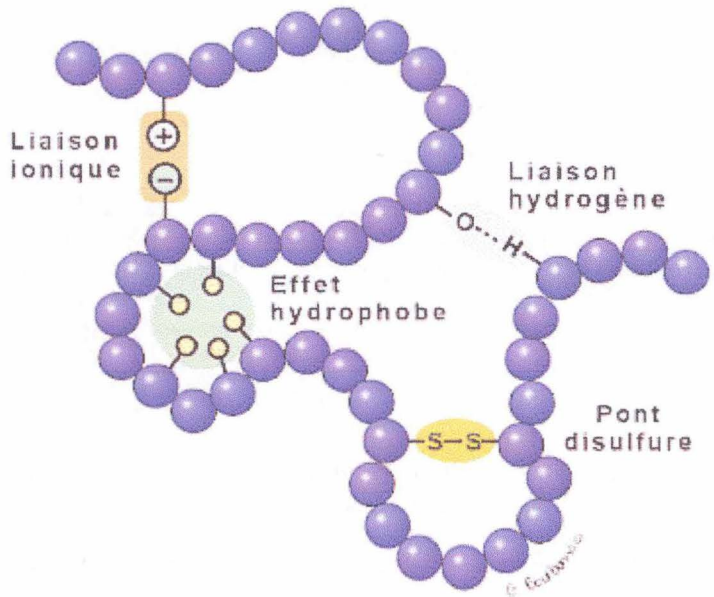


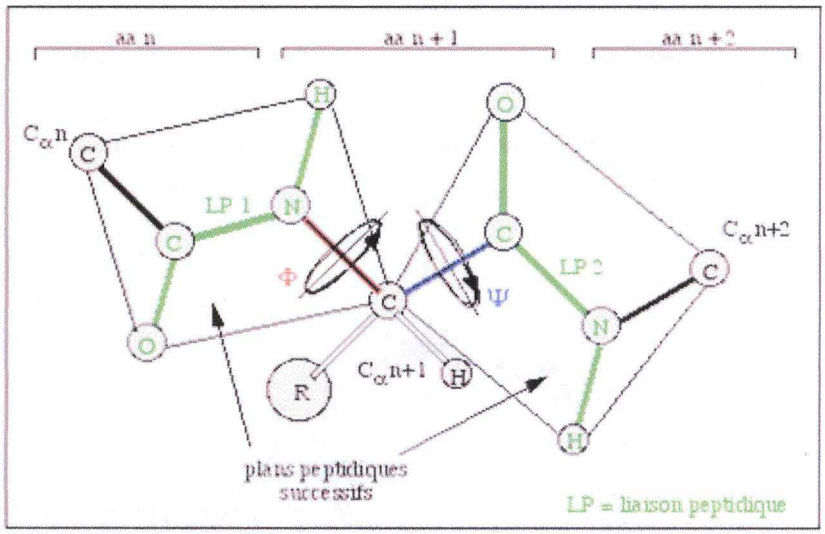
Figure 41 - Structure secondaire

On retrouve dans ce repliement des protéines un certain nombre de motifs communs qui sont les formes caractéristiques de la structure secondaire.

Repliement

La protéine peut se replier sur elle-même, car la structure primaire n'est pas rigide, elle possède plusieurs degrés de liberté.

- Chaque liaison peptidique et ses deux C_α adjacents constituent un bloc rigide situé dans un même plan.
- Chacun des deux plans rattachés à un C_α peut effectuer une rotation libre autour de cette liaison. Ces rotations sont caractérisées par les angles de rotation Φ (phi) et Ψ (psi).



Chaque plan comprend six atomes. Les plans sont articulés entre eux autour des carbones alpha par libre rotation : angle phi (Φ, Cα-N) et psi (Ψ, Cα-C) du même aa.

Figure 42 - Repliement d'une protéine

Les formes que l'on a identifiées et qui constituent la structure secondaire des protéines sont principalement des hélices alpha et des feuillets bêta (tournants et brins bêta).

Les hélices alpha sont des zones ou les acides aminés s'enroulent en hélice. Ces hélices sont stabilisées par des ponts hydrogène.

Les feuillets plissés bêta sont des segments de chaînes (brins bêta) qui se disposent parallèlement les uns aux autres (tournants bêta) et qui forment une structure aplatie, plissée : le feuillet bêta.

Il existe également des zones de structure non définie.

- La duplication, lorsqu'un fragment d'ADN est copié et inséré à une autre localisation chromosomique.
- La délétion, lorsqu'un morceau de chromosome disparaît.
- La translocation, lorsqu'un fragment de chromosome est extrait de son emplacement et inséré, soit ailleurs dans le même chromosome, soit vers une zone d'un autre chromosome.

Phylogénie moléculaire

Elle consiste à utiliser des méthodes informatiques pour analyser des gènes et retracer l'histoire de l'évolution [43] [205].

Le séquençage d'un gène ou d'une protéine chez des organismes appartenant à des espèces plus ou moins proches permet de détecter les différences qui révèlent des mutations apparues au cours de l'évolution, provoquant la divergence des deux espèces.

En effectuant des alignements multiples, on met en évidence les positions conservées et les positions variables. On connaît ainsi le résultat de l'évolution et on peut apprécier la distance entre chaque paire de séquences en comptant le nombre de résidus (nucléotides ou acides aminés) qui diffèrent.

On peut ensuite inférer un arbre phylogénétique en regroupant les séquences de plus grande homologie, en évaluant le nombre de mutations nécessaires pour passer d'une espèce à une autre. Cette distance est fonction de l'arbre que l'on construit ; une technique appliquée est la loi de parcimonie qui essaye d'arriver à construire des chemins qui minimisent le nombre de mutations de transition.

La phylogénie moléculaire constitue un outil puissant pour inférer le degré de proximité de différentes espèces sur base de critères directement liés à leur évolution. Elle est un complément aux méthodes d'anatomie comparée, et se substitue à elles lorsque ces dernières, ou la biochimie ne peuvent donner de critères décisifs de classement, comme par exemple pour les bactéries.

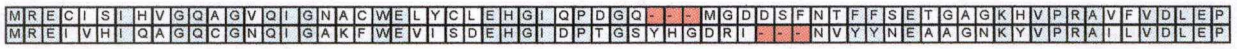
6. LES MATRICES DE SCORES

6.1. Alignements des protéines

Aligner deux ou plusieurs séquences, revient à les placer dans un tableau, une séquence par ligne. Les acides aminés qui sont comparables entre les différentes séquences se retrouvent dans la même colonne du tableau.

Les séquences comparées ont rarement la même longueur. Les insertions et les délétions d'acides aminés ont déplacé sur les différentes protéines les positions relatives des séquences homologues. L'alignement oblige à faire des discontinuités, des trous dans certaines séquences pour pouvoir mettre en correspondance des zones comparables mais séparées par un intervalle différent sur chacune des séquences. Des zones vides dans un alignement sont appelées des gaps et représentées par un tiret.

Ci-dessous, un fragment d'une protéine, la tubuline, avec en bleu les zones d'identité, en rouge des gaps, et en blanc, les zones alignées correspondant à des acides aminés qui ont été substitués.



Ci-dessous un exemple d'alignement de séquences, tel qu'enregistré dans la banque de données Homstrad (ici les séquences ont la même longueur et il n'y a pas de gap) :

```
C; family: spider toxin
C; class: small disulphide
>P1;1iva
structureN:1iva:    1 : :  48 : :omega-agatoxin-IVA:Agelenopsis aperta:-
1.00:-1.00
KKKCIAKDYGRCKWGGTPCCRGRGCICSIMGTNCECKPRLIMEGLGLA*
>P1;1oma
structureN:1oma:    1 : :  48 : :omega-agatoxin-IVB:Agelenopsis aperta:-
1.00:-1.00
EDNCIAEDYGKCTWGGTKCCRGRPCRCSMIGTNCECTPRLIMEGLSFA*
```

Figure 44 - Format de la banque de données HOMSTRAD

6.1.1. Impact de l'évolution

Aligner des séquences d'acides aminés de protéines, c'est mettre en correspondance des séries d'acides aminés de ces protéines, qui sont identiques ou comparables. Elles sont comparables, soit parce qu'elles proviennent d'un ancêtre commun, soit qu'elles correspondent à une même fonction.

En plus des séquences d'acides aminés identiques, l'alignement va mettre en correspondance des acides aminés différents qui se retrouvent à des positions où, soit ils s'intègrent sans modifier la fonction, soit ils sont à des positions non fonctionnelle de la protéine.

La structure tertiaire

Lorsque l'on prend dans son ensemble toute la protéine, et que l'on examine la disposition dans l'espace de toutes les zones de la structure secondaire, la représentation complète de toute la protéine est sa structure tertiaire.

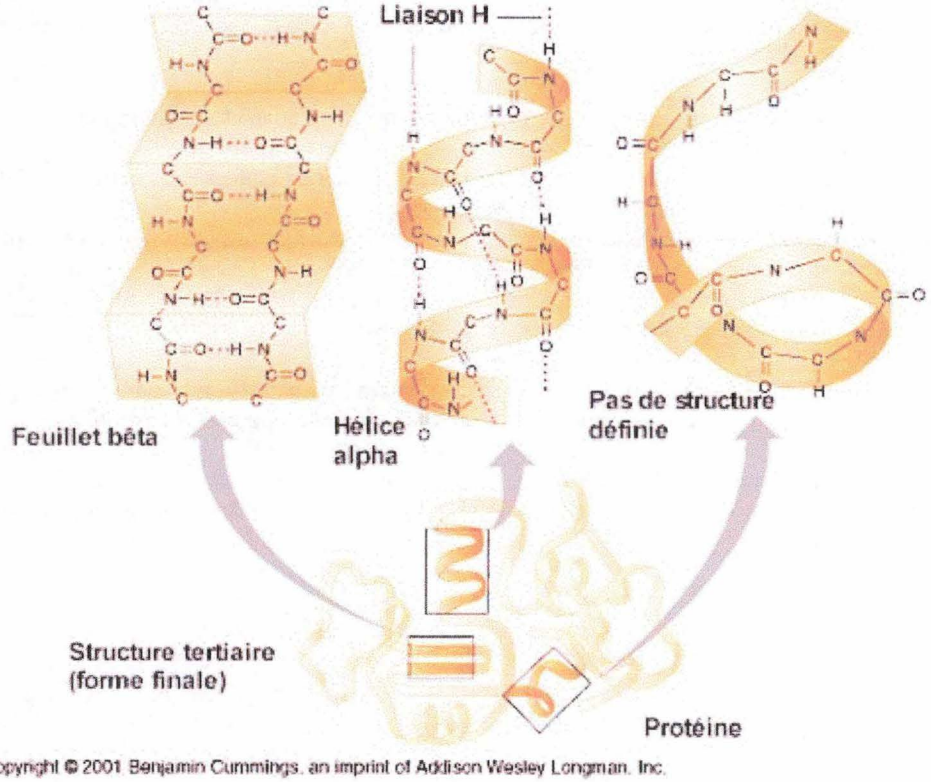


Figure 43 - Structure tertiaire

La structure tertiaire se rapporte aux relations dans l'espace des différentes structures secondaires, hélices et feuillets.

La structure quaternaire

La structure quaternaire concerne les protéines qui contiennent plus d'une chaîne polypeptidique, ce qui nécessite un niveau supplémentaire d'organisation.

5.3. Processus évolutifs et phylogénie moléculaire

L'évolution

Les populations d'individus évoluent génétiquement dans le temps, en fonction des conditions particulières auxquelles elles sont soumises [156] [228]. On distingue:

- La sélection naturelle est un changement évolutif adaptatif ; l'adaptation au milieu donne à l'individu une probabilité plus grande de survivre et de transmettre son génome.
- La dérive génétique est l'effet de glissement dans des populations isolées vers certains facteurs génétiques existants qui sont plus sélectionnés que d'autres.
- Les mutations aléatoires qui produisent un changement de la structure du génome. Elles n'ont un impact sur l'évolution que si elles touchent les « loci » codants.
- Le flux génétique est un brassage qui tend à contrebalancer les tendances évolutives et à homogénéiser la composition génétique des populations.

La phylogénétique étudie l'évolution et la diversification des espèces ; partant d'un ancêtre commun, elle tend à établir la topologie d'un arbre nécessitant le moins grand nombre de changements évolutifs.

Le génome humain comporterait 1,5 % de zones codant pour des protéines et 3,5 % de zones opérationnelles produisant des mécanismes cellulaires (ARN). Les 95 % complémentaires sont des zones non opérationnelles.

Les mutations

Les mutations sont des modifications dans le génome, qui interviennent soit spontanément, soit sous l'action de radiations ou d'agents chimiques. Les mutations sont conservées par l'hérédité. Cependant, lorsqu'elles interviennent dans une zone opérationnelle, elles rendent en général l'individu moins bien adapté et souvent inapte à vivre, tandis que si elles touchent une zone non opérationnelle, elles n'ont aucun impact sur les mécanismes biologiques de l'individu. Le taux de mutation conservé dans le génome est donc plus important dans les zones non opérationnelles.

Les mutations peuvent toucher une zone très localisée d'un gène selon trois modes :

- La substitution d'un nucléotide par un autre.
- L'insertion d'un nouveau nucléotide dans une séquence.
- La délétion d'un nucléotide de la séquence.

Des réarrangements beaucoup plus importants peuvent aussi se produire dans le génome et toucher des fragments très long d'ADN. Ce sont :

Des gènes dupliqués chez un même individu, sont des gènes paralogues, des gènes correspondants chez des individus dont les populations ont évolué et se sont différentiées, sont des gènes orthologues.

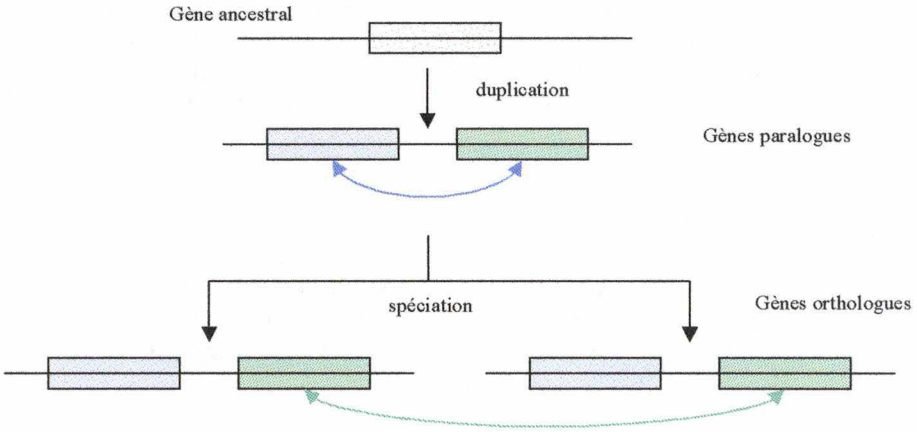


Figure 45 - Différentiation des gènes

Les fonctions des protéines produites par les gènes qui ont muté sont conservées ou très voisines, malgré un nombre parfois élevé de délétions, insertions ou remplacements d'acides aminés.

6.1.2. Impact des propriétés physico-chimiques

Les acides aminés qui se substituent à d'autres dans une protéine, doivent s'intégrer dans un emplacement où ils sont soumis à des contraintes stériques et à des forces électrostatiques. Il y a une plus grande probabilité pour que ces substitutions se réalisent entre acides aminés ayant des propriétés physico-chimiques comparables [2] [33].

Les acides aminés se répartissent en trois catégories: ceux qui sont chargés électriquement, ceux qui ont une polarisation de charge et les acides aminés hydrophobes non polaires.

Deux autres caractéristiques importantes sont les angles Φ (phi) et Ψ (psi) que fait le C_α avec les deux liens peptidiques. Le graphique de Ramachandran donne la répartition des angles correspondant à des structures secondaires stables.

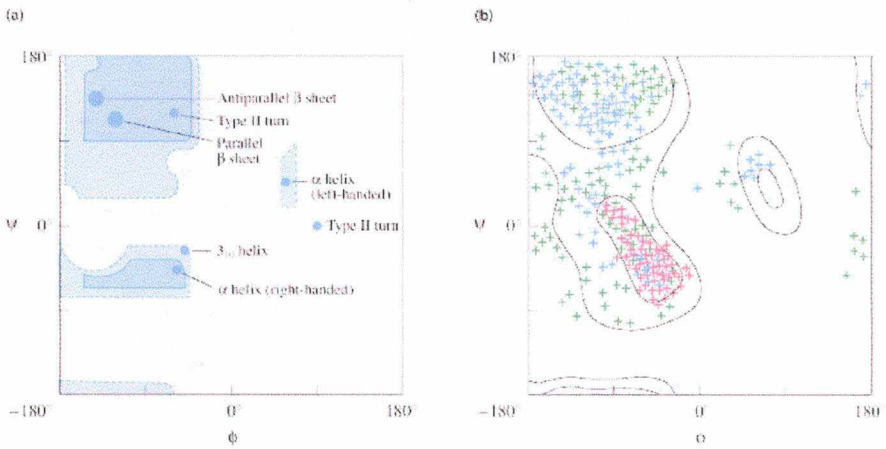


Figure 46 - Propriétés physico-chimiques des acides aminés

6.2. Les matrices de substitution

Ce sont des tables qui codifient une probabilité pour qu'un acide aminé puisse être substitué par un autre dans une séquence protéique. Ces tables contiennent des scores qui permettent d'attribuer un coefficient de similitude ou de distance lors de la comparaison des acides aminés de deux séquences. Plus le score que l'on va établir dans cette comparaison est élevé, et plus les deux acides aminés sont similaires. On distingue plusieurs types de matrices de substitution:

La matrice unitaire ou matrice identité

C'est une matrice qui permet de déterminer le pourcentage d'identité de deux séquences. Le score est 1 si les deux acides aminés sont identiques dans les deux séquences et 0 sinon. C'est la matrice unité, qui alloue un poids identique à la conservation de tous les acides aminés. Cette matrice est peu applicable pour les protéines où l'utilisation de scores différents est nécessaire.

La matrice construite à partir du code génétique

Fitch (1966) établit une matrice de substitution basée sur le nombre minimum de changements nécessaires au niveau des nucléotides pour passer d'un acide aminé à un autre. C'est la « minimum mutation matrix » . Cette matrice ne prend pas en compte la sélection qui ne va conserver qu'une partie des mutations qui peuvent apparaître au cours de l'évolution.

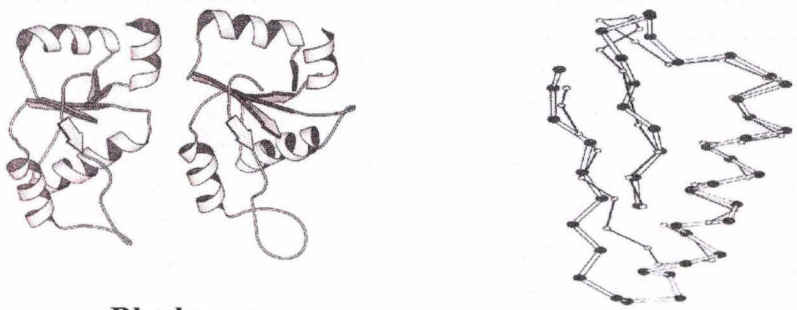
Les matrices construites sur les propriétés physico-chimiques

Les potentialités de substitution entre acides aminés vont être supérieures lorsque leurs propriétés physico-chimiques seront plus proches. Lewit (1976) établi une matrice basée sur l'hydrophobicité des acides aminés (chargé – polaire – non polaire).

Les matrices construites sur les structures 3D

Les données expérimentales sur les structures tridimensionnelles ne cessent de croître, ce qui a permis d'établir des matrices basées sur des comparaisons de structures permettant de comparer des protéines très éloignées. Johnson et Overington (1993) ont établi une telle matrice à partir de 235 structures protéiques réparties en 65 familles. D'autres matrices ont été constituées (Blake, Risler, Johnson).

Alignement de structures 3D



Rhodanese

Source : Risler 2003

Figure 47 - Alignement de structures 3D

Les matrices empiriques déduites de l'évolution :

Elles sont basées sur des alignements de protéines homologues et pour chaque paire d'acides aminés, on calcule le quotient : $R_{ij} = \frac{q_{ij}}{p_i p_j}$

Avec :

q_{ij} fréquence de substitution observée d'un $(AA)_i$ par un $(AA)_j$ et réciproquement.

p_i fréquence de l'acide aminé $(AA)_i$ dans la famille de protéine.

$p_i p_j$ fréquence de substitution d'un acide aminé par l'autre, si les substitutions se faisaient au hasard.

Un quotient supérieur à l'unité indique que les substitutions ont été favorisées par l'évolution.

La matrice de score est construite avec les logarithmes en base deux du quotient R et d'un coefficient de normalisation.

Un score positif indique que la substitution des deux acides aminés est plus fréquente que si elle était due au simple hasard ; un score négatif indique que la substitution réellement observée au cours de l'évolution est moins fréquente que si elle avait été due au hasard seul.

Matrices PAM

Margareth Dayhoff (1978) sélectionne 71 familles et réalise l'alignement de 1300 protéines de forte identité. Elle établit l'arbre phylogénétique de chaque famille et la reconstitution des séquences ancestrales selon le principe de parcimonie. Elle détermine le nombre de substitutions qui permettent de passer des séquences ancestrales aux séquences observées.

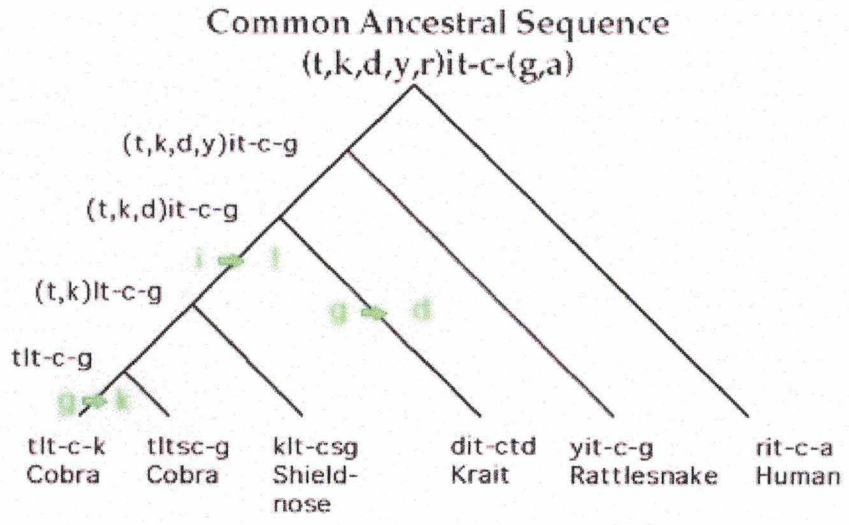


Figure 48 - Matrices PAM - Arbre phylogénétique

La matrice de substitution normalisée est la matrice PAM1. Elle donne les scores de similarité qui correspondent à une mutation pour une séquence de cent acides aminés (*Percent Accepted Mutation*). Cela correspond à un très faible taux de changement, c'est à dire que les séquences doivent être presque identiques pour utiliser cette matrice.

La table de scores est construite avec des valeurs de probabilités logarithmiques multipliées par 10 000.

On dérive à partir de la matrice PAM1, d'autres matrices correspondant à des distances plus éloignées, donc acceptant N mutations pour cent acides aminés. Plus N est élevé, et plus la matrice est susceptible de refléter l'évolution dans des protéines parentes éloignées.

Pour PAM250, on arrive dans la zone d'ombre (*twilight zone*), où l'homologie est de 15% à 25%, c'est à dire qu'elle pourrait être due au hasard.

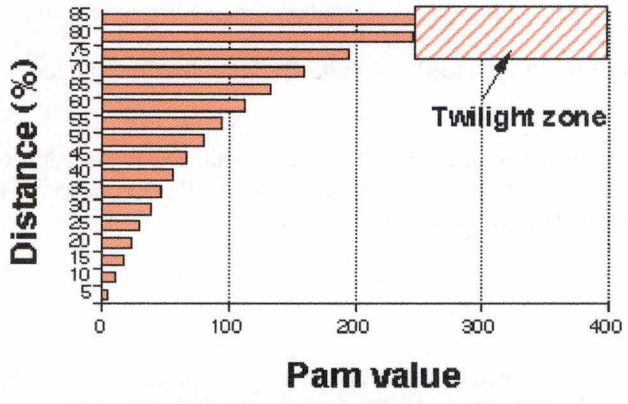


Figure 49 - Matrices PAM - Distances d'évolution

Matrices BLOSUM

Etablies par Henikoff et Henikoff en 1992, à partir de segments courts, très conservés dans des protéines homologues alignées sans gap (*Blocks Substitution Matrix*) [115] [116].

On regroupe des segments en fonction de leur degré d'identité, et pour chaque regroupement on calcule une matrice de substitution correspondante (ex : BLOSUM 80 correspond à des séquences partageant au moins 80% d'acides aminés identiques) - (*log odds matrix*).

Il y a des relations empiriques entre les matrices PAM et BLOSUM, en fonction du degré d'homologie (BLOSUM 62 ⇔ PAM160).

Lors des alignements de séquences, BLOSUM donne en général de meilleurs résultats que PAM, car elle est basée sur des alignements locaux, alors que PAM inclus des régions très divergentes. De plus elles sont établies sur une base d'informations plus large.

Matrice de Gonnet (1992)

C'est une matrice unique basée sur un échantillon très large de 8 344 353 acides aminés de la banque Swiss Prot ou chaque séquence est comparée à l'ensemble des séquences de la banque. Le calcul est fait par itérations jusqu'à convergence.

Tous les alignements significatifs recensés servent ensuite à générer une matrice avec une distance PAM de 250.

Le coût du gap est évalué par :

$$10 \ln(P) = -36{,}71 + 7{,}44 \ln(\text{dist PAM}) - 14{,}92 \ln(k)$$
$$10 \ln(P) = -20{,}63 - 1{,}65 (k-1)$$

Pondération des gaps

Le score d'un gap (S_p) vient modifier le score calculé (S_c) pour les acides aminés en correspondance.

$$\text{Score} = \Sigma [S_c]_{sim} - \Sigma [S_p]_{ins} - \Sigma [S_p]_{del}$$

Le score est d'autant plus élevé que la zone de similitude considérée est longue. D'autre part, selon l'importance et la pondération que l'on donne aux insertions et aux délétions, on peut nuancer le résultat et favoriser une situation plutôt qu'une autre.
On peut donner une pénalité fixe ou une pénalité variable, selon la longueur du gap.

Choix d'une matrice

Le choix de la matrice peut influencer ou biaiser le résultat. Les différentes études montrent qu'il n'existe pas de matrice idéale. Elles mettent en évidence que la matrice PAM 250 de Dayhoff donne un poids trop important à l'identité et est mal adaptée à la comparaison de protéines distantes car elle ne comporte pas suffisamment d'informations structurales.

Les matrices plus récentes sont supérieures aux matrices PAM de Dayhoff.

Jones et al. (1992) ont déterminé de nouvelles matrices PAM, avec des données plus récentes pour obtenir des résultats comparables à des matrices comme BLOSUM 62 ou BLOSUM 50.

Selon une étude de Vogt et al. (1995), toutes les matrices donnent de meilleurs résultats pour des alignements globaux que pour des alignements locaux et leurs performances sont très sensibles aux modalités de pénalités choisies pour tenir compte des gaps. Il montre également, que tenant compte de différentes combinaisons d'algorithmes, de matrices et de choix de pénalité, c'est la matrice de Gonnet qui donne le meilleur résultat. Viennent ensuite les matrices BLOSUM 62 et BLOSUM 50, les matrices de structure de Overington et une matrice de Brenner (1994), semblable à celle de Gonnet.

Les méthodes de recherche dans les banques de données

Diverses méthodes d'alignement sont utilisées pour comparer une séquence donnée aux séquences contenues dans les banques de données [24] [249].

Matrice de points « DOTPLOT »

Les deux séquences à aligner sont placées le long des axes d'un graphique. On marque d'un point l'intersection d'une ligne et d'une colonne lorsque la même lettre (acide aminé) se retrouve dans les deux séquences. La similarité entre séquences se traduit par une suite de points alignés.

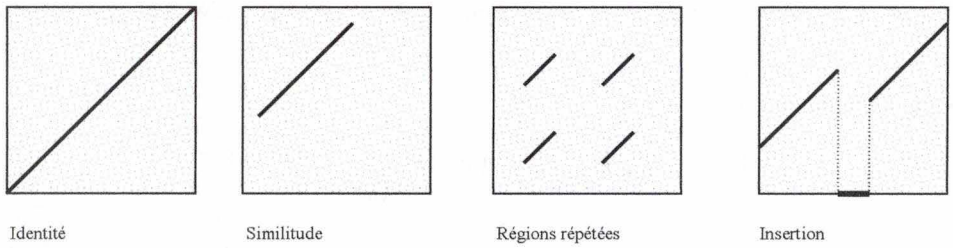


Figure 50 - DOT PLOT - matrice de points

Certains points génèrent du « bruit » car ils sont en correspondance statistique sans être porteur de similitude. Pour les filtrer on utilise des fenêtres de comparaison que l'on fait glisser le long des séquences. On ne met un point de similitude que si la correspondance de la fenêtre dépasse un score seuil fixé.

Exemple de matrice *dot plot* pour une protéine de la drosophylle présentant de nombreuses séquences répétitives (comparaison de la protéine à elle-même).

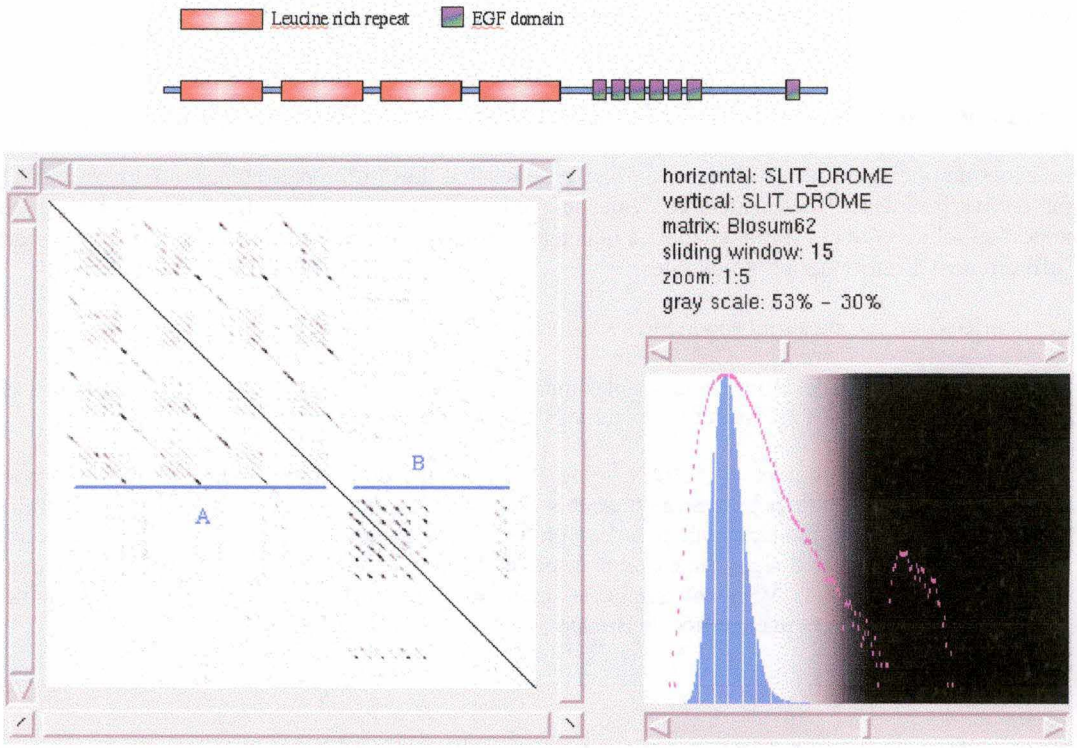


Figure 51 - DOTPLOT - Exemple pour la drosophylle

Source : http://www.isrec.isb-sib.ch/java/dotlet/repeats.html

6.3. Les méthodes d'alignement

Elles sont basées sur une approche statistique ou sur de la programmation dynamique

- Algorithme de Needleman et Wunch

  C'est une méthode de programmation dynamique, qui permet de trouver un alignement optimal global entre deux séquences.
  (Needleman et Wunch. J. Mol.Biol. 488 : 433-453 (1970)) [188].

- Algorithme de Smith et Waterman

  C'est une méthode de programmation dynamique. Elle permet de trouver l'alignement optimal local entre deux séquences.
  (Smith et Waterman . J.Mol. Biol. 25(147) : 195-7 (1981)) [222].

- Algorithme Match-Box

  C'est une méthode statistique, qui permet de réaliser des alignements multiples de sequences de proteines.
  (Depiereux, E. et Feytmans, E. - Match-Box, a fundamentally new algorithm for the simultaneous alignment of several protein sequences, Comput. Appl. Biosci. 8(5) : 501-509 (1992)) [79].

Recherche dans les bases de données

Les principales banques de données protéiques sont Swiss-Prot, EMBL, PIR, HOMSTRAD. La recherche dans une banque se fait généralement par une demande, que l'on soumet à un serveur. On présente une séquence à analyser, et on peut préciser les paramètres de la recherche.

Le serveur va utiliser un programme d'alignement de séquences, pour identifier dans la banque de données des séquences homologues à la séquence protéique qu'on lui a soumis. Il va comparer la séquence fournie par l'utilisateur à toutes les séquences de la banque.

Cette recherche peut être effectuée par des algorithmes de programmation dynamique ou par des méthodes statistiques. Ces méthodes sont rigoureuses mais nécessitent des ressources informatiques conséquentes.

Les serveurs utilisent le plus souvent des méthodes heuristiques très rapides qui permettent une recherche de forte similarité entre segments, sans pour cela garantir que l'alignement soit optimal.

Les deux programmes les plus courants sur les serveurs sont BLAST et FASTA, et leurs variantes :

- BLAST : *Basic Local Alignment Search Tool*

  - La protéine analysée, proposée au serveur, est découpée en mots de longueur fixe. Par défaut la longueur est de 3 acides aminés (20 x 20 x 20 = 8000 possibilités).
  - On construit des synonymes de ces mots, en se fixant un seuil d'acceptation pour le score (matrice BLOSUM 62).
  - On recherche des mots identiques dans la banque de données, puis on fait grandir ces embryons de séquences pour constituer des HSSP (*High Scoring Segment Pair*).
  - On conserve les meilleurs HSSP et on réalise un alignement local optimal par l'algorithme de Smith et Waterman.

- FASTA :

  - On recherche des mots identiques de k lettres entre les séquences (protéine analysée par rapport aux séquences de la banque de données) ; par défaut le mot est de longueur de deux acides aminés.
  - Les mots situés sur une même diagonale de la matrice « *dot plot* » et à une distance inférieure à un seuil fixé sont réunis, ainsi que les régions qui les séparent.
  - On recherche les diagonales les plus significatives sur base d'une matrice de score (PAM 250, BLOSUM 62).
  - On définit des régions proches.
  - On réalise un alignement optimal local sur base de l'algorithme de Smith et Waterman.

Programmes d'alignement multiple

Programme d'alignement de séquences développé par l'unité de recherche en Biologie moléculaire de l'Université de Namur (serveur FUNDP).

- MATCH-BOX :

  Ce programme d'alignement se déroule en trois étapes :

  - *SCANNING* : on définit une fenêtre de longueur W que l'on fait glisser sur la protéine à analyser et on la compare à tous les segments de même longueur sur les autres protéines du groupe de comparaison ;

    On calcule une distance entre tous les segments, sur base de scores entre les acides aminés présents dans les fenêtres.

    On modifie de manière aléatoire l'ordre des acides aminés des séquences pour déterminer un bruit de fond statistique (seuil d'acceptabilité = borne supérieure des distances acceptées).

  - MATCHING : partant du début de chaque séquence et des distances les plus courtes, on associe les fenêtres, en sélectionnant sur chaque séquence de comparaison, la première fenêtre rencontrée qui répond au critère de distance.

On constitue ainsi des groupes appelés boites, dont on valide les distances réciproques de chaque élément par rapport à un seuil, avant de les accepter.

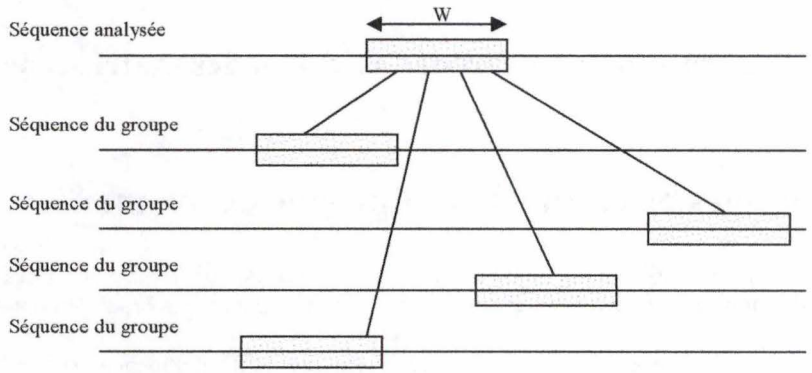


Figure 52 - "MATCH-BOX" - validation des distances entre boites

On constitue par itération, toutes les boites de distance, qui correspondent à un seuil fixé

- SCREENING : c'est l'étape d'alignement

  On compare les boites obtenues et on associe les boites contiguës pour former des boites de plus grande taille.

  On réalise ensuite l'alignement à partir de la séquence la plus longue et on recherche à droite et à gauche des boites compatibles pour un autre alignement.

Autres programmes

Il existe de nombreuses autres méthodes pour lesquelles on peut se référer à l'étude réalisée par Christophe Lambert dans sa thèse : « Développement d'une méthode automatique fiable de modélisation de la structure tridimensionnelle des protéines par homologie et application au génome de *Brucella melitensis* » - Presses Universitaires de Namur (2003) [80] [142] [143].

Nous développerons des éléments plus spécifiques, liés aux travaux d'application réalisés pour ce mémoire, dans le chapitre sept.

Certaines méthodes plus spécifiques sont basées sur la reconnaissance de profils (« *pattern* ») ou sur des alignements de structures [120] [124] [126] [129] [179] [202] [232].

7. APPLICATION

Application des réseaux de neurones au calcul des matrices de scores

7.1. Avantages attendus de l'approche expérimentée

Le premier objectif de ces applications est de rechercher une nouvelle voie pour réaliser des alignements de séquences d'acides aminés afin de retrouver des protéines similaires dans des bases de données.

La question posée est la suivante: « Si on présente à un système d'intelligence artificielle une base de données de protéines alignées, sera-t-il capable d'apprendre tous les critères d'alignement pour les appliquer. Lorsqu'on l'interroge ensuite en lui présentant une protéine inconnue, saura-t-il sélectionner dans une base de données de protéines connues, toutes les protéines de cette base de données qui peuvent être alignées à cette protéine qu'on lui soumet ? »

Dans quelle mesure un système d'intelligence artificielle va-t-il permettre de répondre totalement ou partiellement à cette question ? L'apport de l'approche expérimentée est de mieux cerner les limites de faisabilité d'un tel système. En effet, les théories développées nous livrent des formules générales dont l'application dans des systèmes particuliers très complexes doit être validée par l'expérimentation.

Le second objectif est la formation à l'outil par l'expérimentation. Nous avons réalisé une programmation personnelle d'un réseau de neurones sur base des formules de transfert direct à chaque nœud et des formules de la rétro propagation (voir annexes B et C). Nous avons également utilisé des fonctions du logiciel MATLAB.

7.2. Procédures et résumé des tests

7.2.1. Cadrage de l'expérimentation

Le chapitre 6 décrit les principales méthodes d'alignement qui sont actuellement d'application.

La réalisation d'un alignement complet de deux protéines ne peut s'effectuer uniquement et entièrement par un réseau de neurones. Des algorithmes tels que BLAST, FASTA ou MATCH-BOX ont une logique qui dépasse le domaine d'application des réseaux de neurones.

Malgré leur complexité, ces algorithmes ne se suffisent cependant pas à eux-mêmes pour réaliser un alignement. Ils ont besoin d'un prétraitement statistique qui leur donne la probabilité que deux acides aminés ont de se retrouver « alignés », c'est à dire en correspondance l'un par rapport à l'autre, lorsque les deux protéines que l'on considère sont déclarées alignées. Ces algorithmes utilisent des matrices de scores. Ces matrices sont différentes car, pour les construire, diverses hypothèses ont été faites sur l'évolution du vivant. Il en résulte que le résultat obtenu pour une recherche d'alignement est influencé par le choix de la matrice que l'on utilise pour réaliser cet alignement.
Notre objectif est d'entraîner un réseau de neurones pour qu'il construise lui-même des matrices de scores, à partir de bases de données existantes. Dans notre cas, la base de données qui sera utilisée comme « base

de connaissance » est la base de données HOMSTRAD. Elle est constituée d'un ensemble de protéines connues dont la correspondance de structure a été déterminée par des méthodes physiques (radio cristallographie, résonance magnétique), et logique (homologie entre protéines).

La matrice de scores ainsi construite pourra alors être utilisée par des algorithmes d'alignement (par exemple MATCH-BOX). L'intérêt est de construire ces matrices de score de manière dynamique, sur des ensembles de données qui peuvent être spécifiques d'un domaine particulier que l'on étudie, ou générales si on englobe toutes les familles de protéines.

Les réseaux de neurones réalisent un apprentissage supervisé. Ils ne peuvent reconnaître que ce qu'ils ont appris. Nous allons donc devoir construire un réseau de neurones et l'entraîner pour lui apprendre à construire un modèle statistique, sur la base duquel sera construite une matrice de scores.

7.2.2. Description de la méthode suivie

Notre expérimentation concerne la construction d'une matrice de scores utilisable par des algorithmes existants (chapitre 6).

Nous avons d'abord examiné la granularité de la matrice à construire. Toutes les matrices actuelles sont de granularité unitaire, où un acide aminé unique est mis en correspondance avec un autre acide aminé. C'est également la solution que nous avons retenue. La problématique du choix d'une granularité supérieure, c'est-à-dire des alignements de doublets ou de triplets est traitée à l'annexe F. Une matrice de scores de granularité deux a été calculée.

Environnement de référence

Pour contrôler la pertinence des résultats qui seront obtenus avec les réseaux de neurones, nous avons construit la matrice de scores de référence de la base de données HOMSTRAD. Elle a été testée et validée avec un module de l'application MATCH-BOX.

La matrice de référence a ensuite été utilisée pour réaliser l'apprentissage d'un réseau de neurones. Le réseau construit apprend les 441 points de la matrice de scores et les restitue lorsqu'il est interrogé.

Prototype de réseau de neurones développé avec une base de données de taille réduite

La construction par un réseau de neurones d'une matrice de scores à partir de la base de donnée d'alignements de structures HOMSTRAD nécessite de traiter des fichiers de taille considérable. Nous avons construit une base de données de taille réduite pour réaliser la mise au point de l'application (voir annexe D).

Un prototype de réseau de neurones a été développé (voir annexe E) et entraîné sur cette base de données de taille réduite. La procédure mise au point a permis de faire apprendre par ce prototype les probabilités correctes d'associations de paires d'acides aminés présents dans cette base de données. Ceci a été vérifié par comparaison avec le résultat obtenu par un calcul statistique. On peut en conclure que le réseau de neurones « apprend » lors de son entraînement les éléments de base du calcul de la matrice de scores. La matrice elle-même est calculée par la suite sur base de ces éléments.

Application à la base de données HOMSTRAD

Calcul de probabilités sur base d'un échantillon

Nous avons ensuite appliqué la méthode développée sur la base de données réduite à un échantillon extrait de la base de données HOMSTRAD.

Le réseau de neurones apprend les probabilités d'associations des paires d'acides aminés présents dans l'échantillon avec une erreur inférieure à un centième de pourcent, ce que nous avons vérifié en construisant la statistique de l'échantillon.

L'échantillon tiré pour le test présente un biais par rapport à l'information contenue dans la base de données prise dans son entièreté. Le contrôle effectué par un calcul statistique met en évidence une erreur de un à dix pourcents de la composition de l'échantillon par rapport à la composition effective de la base de données. Les erreurs importantes portent sur les paires d'acides aminés peu représentées, dont la probabilité est très faible par rapport aux paires d'acides aminés identiques.

Les probabilités d'association des acides aminés sont les éléments de base pour le calcul des scores. Notre matrice de scores de référence a été établie sur base d'une statistique exploitant la totalité de la base de donnés « HOMSTRAD ». Nous pouvons donc mesurer la validité des résultats obtenus avec les réseaux de neurones, en évaluant la validité des probabilités apprises par ces réseaux, en comparaison avec les probabilités de référence établies sur la base de données HOMSTRAD prise dans sa totalité.

Calcul de scores d'affinité

Le réseau de neurones a ensuite été utilisé pour construire une matrice de scores basée sur un regroupement des acides aminés, en fonction de leurs propriétés physico-chimiques.

7.3. Expérimentation et résultats

7.3.1. Environnement de référence

7.3.1.1. *Construction d'un environnement de référence pour valider les résultats obtenus par les réseaux de neurones*

L'objectif de ce travail est de faire calculer par un réseau de neurones une matrice de scores qui pourra être exploitée par l'application MATCH-BOX, développée dans l'Unité de Recherche en Biologie Moléculaire (URBM) des facultés universitaires de Namur (Académie Louvain), pour réaliser des alignements multiples de protéines. L'application MATCH-BOX sera aussi utilisée pour valider la matrice qui sera construite, en comparant les résultats d'alignement obtenus avec cette matrice, à des résultats connus obtenus avec les autres matrices de scores de la littérature.

Les recherches bibliographiques que nous avons réalisées mettent en évidence qu'il s'agit d'une voie nouvelle, car aucune publication dans la littérature scientifique qui traite des alignements de séquences ne fait état de l'utilisation de matrices de scores, construites par des réseaux de neurones.

Dans une première approche, nous avons développé une application qui construit une matrice de scores de référence, en utilisant les méthodes statistiques qui sont appliquées pour la construction des matrices BLOSUM, en utilisant comme source d'alignements de référence la base de données HOMSTRAD. C'est cette base de donnée qui sera exploitée comme base de connaissance par les réseaux de neurones.

Cette première étape nous permettra d'une part de valider les résultats qui seront produits par les réseaux de neurones, et d'autre part, de tester avec quelle précision les réseaux de neurones vont pouvoir construire et reconnaître une matrice de plus de 400 points de valeurs quelconques, totalement discontinues.

Construction « classique » de la matrice de scores de référence

Nous avons calculé pour l'ensemble de la base de données HOMSTRAD, la probabilité pour qu'un acide aminé soit substitué par un autre acide aminé, et donc que ces deux acides aminés se trouve en correspondance dans deux protéines alignées. Nous avons également calculé la probabilité des gaps dans HOMSTRAD. Un gap signifie qu'il y a eu à cet endroit, une insertion ou une délétion d'acide aminé dans une des deux protéines.

Ces probabilités sont un pur traitement statistique, identique à celui qui est effectué pour le calcul des scores des matrices BLOSUM.

$$\text{Score} = \text{LOG (probabilité observée / probabilité attendue)}$$

Matrice des probabilités de substitution observées

Soit un bloc de n séquences alignées, de longueur L.
Pour chacune des positions de 1 à L, le nombre de paires d'acides aminés à comparer est :

$$C_n^2 = \text{fact } n \,/\, \text{fact } (n-2) * \text{fact } 2 = n * (n-1) / 2$$

La probabilité observée d'avoir une paire ordonnée d'acides aminés (i,j) est :

$Q_{ij} = N_{ij} \,/\, \Sigma\Sigma N_{ij}$ $(1 \leq i,j \leq 21)$

N_{ij} est le nombre de paires dénombrées (20 AA et 1 gap).

Nous avons également $\Sigma\Sigma N_{ij} = C_n^2$

Nous avons d'abord réalisé une matrice carrée lors des comptages, puis nous l'avons transformée en matrice triangulaire en sommant Q_{ij} et Q_{ji}, pour $i \neq j$.

Matrice des probabilités de substitution attendue

Soit N_i, le nombre d'acides aminés de type i pour la totalité des séquences :

$$P_i = N_i \,/\, 2\ C_n^2$$

Soit E_{ij}, la probabilité attendue d'avoir statistiquement une association des acides aminés i et j dans une paire :

Pour $i = j$, $\quad E_{ij} = P_i * P_i$
Pour $i \neq j$, $\quad E_{ij} = P_{ij} + P_{ji} = 2 * P_i * P_j$

Calcul des Odds-ratios

On construit une matrice triangulaire pour les paires d'acides aminés :

$$O_{ij} = Q_{ij} / E_{ij} \text{ pour } 1 \leq i,j \leq 21$$

Matrice des log-Odds

Pour comparer les résultats obtenus avec les matrices BLOSUM, nous avons calculé le log-Odds 1 /2 bit ($\lambda = ½$)

$$S_{ij} = (1/\lambda) \log(Q_{ij}/E_{ij})$$

Nous avons travaillé avec la précision des données S_{ij}, sans les arrondir à l'unité, comme cela est réalisé dans les matrices BLOSUM

$$\text{Score BLOSUM} = \text{ROUND}\ ((1/\lambda) \log (Q_{ij} / E_{ij}))$$

L'échantillon de protéines alignées servant d'exemple de référence dans notre calcul est différent de celui utilisé pour le calcul des scores des matrices BLOSUM, mais la procédure de calcul est identique.

Matrice calculée avec la base de données HOMSTRAD

5	1	-2	-1	-2	0	-2	-1	-1	-1	0	-1	-1	-1	-1	0	0	0	-1	-2	-2
-1	6	-2	-3	-2	-2	-3	-2	-4	-2	-1	-2	-3	-2	-2	-1	-1	-1	-2	-2	-2
-2	-2	6	1	-4	-2	-1	-3	-1	-4	-3	0	-1	0	-1	0	-1	-2	-2	-2	-1
-1	-3	1	5	-2	-2	-1	-3	0	-2	-1	-1	-1	1	-2	-1	-1	-2	-1	-1	-1
-3	-1	-3	-3	6	-2	-1	0	-3	0	0	-2	-2	-1	-2	-2	-3	-2	1	1	-2
0	-2	-1	-2	-4	6	-1	-4	-2	-4	-3	-1	-2	-2	-3	-1	-2	-3	-2	-1	-2
-2	-3	-1	-1	-1	-2	7	-1	0	-1	-1	0	-2	-1	0	-1	-1	-1	-1	0	-1
-1	-1	-4	-2	-1	-4	-2	5	-2	0	1	-2	-2	0	-1	-2	-1	1	-1	-1	-2
-1	-3	-1	-1	-2	-3	-1	-2	5	-1	-1	0	-1	0	1	-1	-1	-1	-2	-2	-2
-2	-1	-4	-2	1	-4	-2	1	-1	5	1	-3	-2	-2	-2	-2	-1	0	-1	-1	-2
0	0	-3	-2	0	-2	-1	1	0	1	6	-1	-3	0	-1	-2	-1	1	0	-1	-1
-1	-2	1	0	-2	0	0	-2	-1	-2	-1	6	0	-1	0	1	0	-3	-2	-2	-1
-1	-3	-1	-1	-3	-2	-1	-2	-1	-3	-1	-1	6	0	-2	0	-1	-2	-2	-2	-1
0	-3	0	1	-2	-2	0	-2	1	-2	-1	-1	-1	6	1	-1	0	-2	-1	-2	-1
-1	-3	-2	-1	-2	-3	0	-2	1	-1	-2	-1	-1	1	5	0	-2	-2	-2	-1	-2
0	-1	0	-1	-2	-1	0	-3	-1	-3	-1	1	0	-1	0	5	2	-2	-1	-2	-1
0	-1	-1	-1	-2	-2	-1	0	0	-1	-1	0	-1	-1	-1	2	5	0	-2	-2	-2
0	-1	-3	-3	-1	-2	-2	2	-2	1	0	-3	-2	-2	-2	-2	0	5	1	-2	-2
-2	-2	-3	-2	1	-2	-1	-1	-3	-1	0	-2	-2	-2	-2	-2	-2	0	9	2	-3
-2	-2	-2	-1	1	-2	0	-2	-2	-1	-1	-1	-2	-2	-2	-2	-2	-2	2	7	-1
-2	-2	-1	-1	-1	-2	-1	-2	-1	-2	-1	-1	-1	-1	-2	-1	-2	-3	-2	-2	3

Figure 53 - Nouvelle matrice de scores ©

La matrice calculée en exploitant la totalité de la base de données HOMSTRAD est assez proche de la matrice BLOSUM 62. La comparaison entre les deux matrices est donnée ci-après.

Cette matrice de scores a été validée avec un module de l'application MATCH-BOX, qui a mis en évidence que les résultats d'alignements que l'on obtient par son utilisation sont supérieurs à ceux obtenus par les matrices « classiques ». Elle constitue donc une référence pour valider les calculs de probabilité ainsi que les matrices de scores qui pourront être produites par un réseau de neurones.

Comparaison de la matrice construite sur HOMTRAD, avec la matrice BLOSUM 62

Matrice BLOSUM 62

2	0	-1	0	-1	0	-1	-1	0	-1	0	-1	0	0	-1	1	0	0	-1	-1	-2
0	4	-2	-2	-1	-1	-1	-1	-2	-1	-1	-1	-1	-1	-2	0	0	0	-1	-1	-2
-1	-2	3	1	-2	-1	-1	-2	0	-2	-2	1	-1	0	-1	0	-1	-2	-2	-2	-2
0	-2	1	2	-2	-1	0	-2	0	-1	-1	0	-1	1	0	0	0	-1	-1	-1	-2
-1	-1	-2	-2	3	-2	-1	0	-2	0	0	-1	-2	-2	-1	-1	-1	0	0	1	-2
0	-1	-1	-1	-2	3	-1	-2	-1	-2	-1	0	-1	-1	-1	0	-1	-2	-1	-2	-2
-1	-1	-1	0	-1	-1	4	-2	0	-1	-1	0	-1	0	0	0	-1	-2	-1	1	-2
-1	-1	-2	-2	0	-2	-2	2	-1	1	1	-2	-1	-1	-1	-1	0	1	-1	-1	-2
0	-2	0	0	-2	-1	0	-1	2	-1	-1	0	-1	1	1	0	0	-1	-1	-1	-2
-1	-1	-2	-1	0	-2	-1	1	-1	2	1	-2	-1	-1	-1	-1	-1	0	-1	-1	-2
0	-1	-2	-1	0	-1	-1	1	-1	1	3	-1	-1	0	-1	-1	0	0	-1	0	-2
-1	-1	1	0	-1	0	0	-2	0	-2	-1	3	-1	0	0	0	0	-1	-2	-1	-2
0	-1	-1	-1	-2	-1	-1	-1	-1	-1	-1	-1	4	-1	-1	0	-1	-1	-2	-1	-2
0	-1	0	1	-2	-1	0	-1	1	-1	0	0	-1	3	0	0	0	-1	-1	-1	-2
-1	-2	-1	0	-1	-1	0	-1	1	-1	-1	0	-1	0	3	0	-1	-1	-1	-1	-2
1	0	0	0	-1	0	0	-1	0	-1	-1	0	0	0	0	2	1	-1	-1	-1	-2
0	0	-1	0	-1	-1	-1	0	0	-1	0	0	-1	0	-1	1	2	0	-1	-1	-2
0	0	-2	-1	0	-2	-2	1	-1	0	0	-1	-1	-1	-1	-1	0	2	-1	-1	-2
-1	-1	-2	-1	0	-1	-1	-1	-1	-1	-1	-2	-2	-1	-1	-1	-1	-1	5	1	-2
-1	-1	-2	-1	1	-2	1	-1	-1	-1	0	-1	-1	-1	-1	-1	-1	-1	1	3	-2
-2	-2	-2	-2	-2	-2	-2	-2	-2	-2	-2	-2	-2	-2	-2	-2	-2	-2	-2	-2	1

Ecart entre les deux matrices

-2	-1	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	-1
0	0	-1	0	0	0	0	1	1	1	0	0	1	0	0	0	0	1	0	0	-1
0	0	-1	0	1	1	0	1	0	1	1	0	0	0	0	0	0	0	0	0	-1
0	0	0	-1	0	0	1	1	0	0	0	0	0	0	1	1	0	0	-1	0	-2
1	0	1	1	-2	0	0	0	1	0	0	0	0	-1	0	0	1	1	0	1	-1
0	0	0	0	1	-1	-1	1	1	1	1	0	1	1	1	1	1	0	0	-1	-1
1	1	0	0	0	1	-1	-1	0	-1	0	0	0	1	0	0	0	-1	-1	1	-1
0	0	1	0	0	1	0	-2	0	1	0	0	0	-1	0	0	0	1	-1	0	0
1	1	0	1	0	1	1	0	-1	-1	0	0	0	1	0	1	0	0	0	1	-1
0	0	1	0	0	1	0	0	0	-2	0	1	0	0	0	0	0	0	0	0	-1
0	0	0	0	0	0	0	0	0	0	-1	0	1	0	0	0	1	0	-1	0	-1
0	0	0	0	0	0	0	0	1	0	0	-1	-1	1	0	0	0	0	-1	0	-1
0	1	0	0	0	0	0	0	0	1	0	0	0	-1	1	0	0	0	0	0	-1
0	1	0	0	0	1	0	0	0	0	1	0	0	-1	0	1	0	0	0	0	-1
0	0	0	1	0	1	0	0	1	0	1	0	0	0	-1	-1	1	0	0	0	-1
0	0	0	0	0	1	0	1	1	1	0	0	0	0	0	-1	-1	0	0	1	-1
0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	-1	0	0	1	-1
0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	-1	-2	1	0
0	0	0	0	0	0	-1	-1	1	0	-1	0	0	0	0	0	0	-1	-1	-1	0
0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	1	0	0	-1	-1
-1	-1	-1	-1	-2	-1	-1	0	-1	0	-1	-1	-1	-1	-1	-1	-1	0	-1	-1	-2

Figure 54 - Comparaison avec la matrice BLOSUM 62

7.3.1.2. *Apprentissage d'une matrice de scores par un réseau de neurones*

L'exemple développé à l'annexe C illustre le fait qu'un réseau de neurones à deux couches permet d'approximer une fonction continue dans R2. On vérifie bien qu'il est un approximateur universel de fonctions continues.

Nous souhaiterions faire reconnaître à un réseau un ensemble de points discontinus que sont les probabilités de substitution de paires d'acides aminés, que nous avons calculé comme indiqué ci-dessus, avec des méthodes statistiques classiques.

Les données d'apprentissage sont les scores de la matrice, soit 20 AA et un gap. Cela représente 441 données. Pour garantir une plus grande sensibilité, les valeurs utilisées pour les scores sont prises avant le calcul des arrondis.

Nous avons transformé la matrice de scores (triangulaire) en une matrice carrée symétrique, dont voici la représentation dans l'espace :

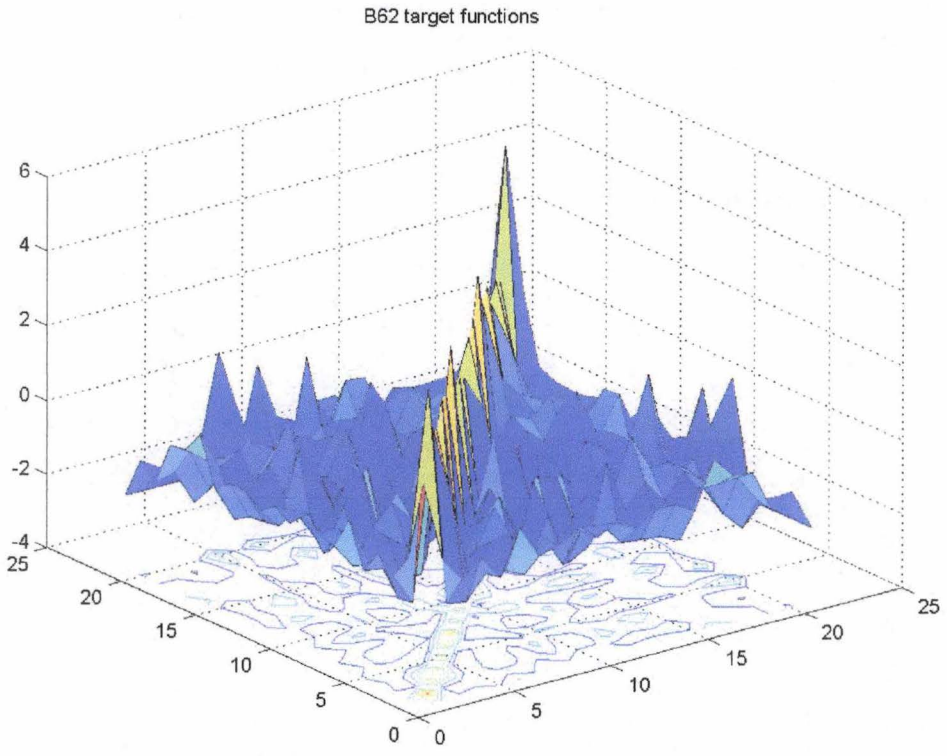


Figure 55 - Fonction discontinue à apprendre par le réseau de neurones

La matrice est symétrique par rapport à la diagonale, car on donne la même probabilité aux paires (i,j) et aux paires (j,i).

La structure du réseau de neurones est identique à celle utilisée pour l'étude de la fonction continue.

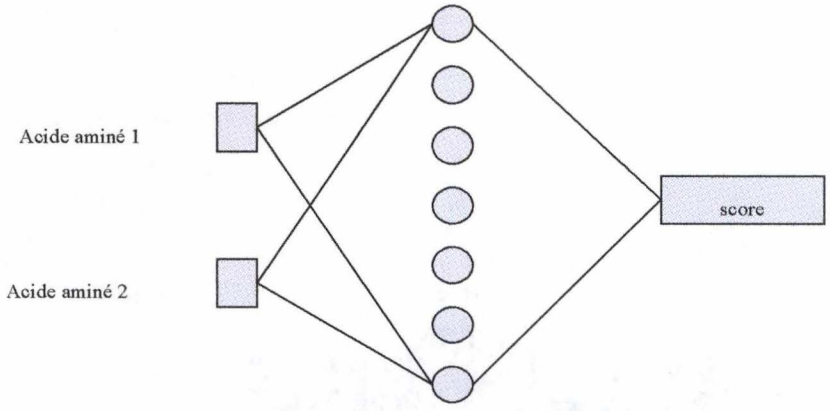


Figure 56 - Structure du réseau utilisé

Caractéristiques :

- Deux neurones dans la couche d'entrée
- Quatre cent quatre vingt neurones cachés
- La fonction de transfert est une tangente hyperbolique
- Un neurone dans la couche de sortie

On réalise un sur-apprentissage, avec la totalité des données de la base d'exemples, qui est ici constituée par les 441 valeurs à faire apprendre par le réseau.

L'apprentissage du réseau est rapide. L'erreur résiduelle descend à 10^{-6}, après 15 périodes d'apprentissage (époques). L'algorithme utilisé est celui de Leverberg Marquardt.

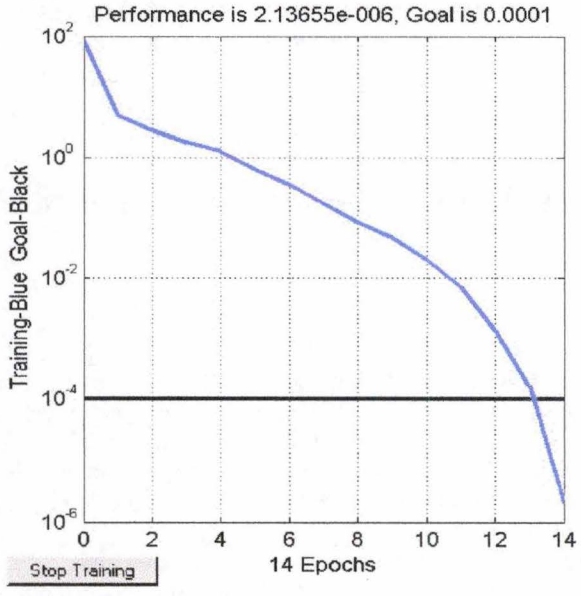


Figure 57 - Evolution de l'erreur au cours de l'apprentissage

Interrogation du réseau

On interroge le réseau pour toutes les paires d'acides aminés. Celui-ci restitue des résultats corrects.

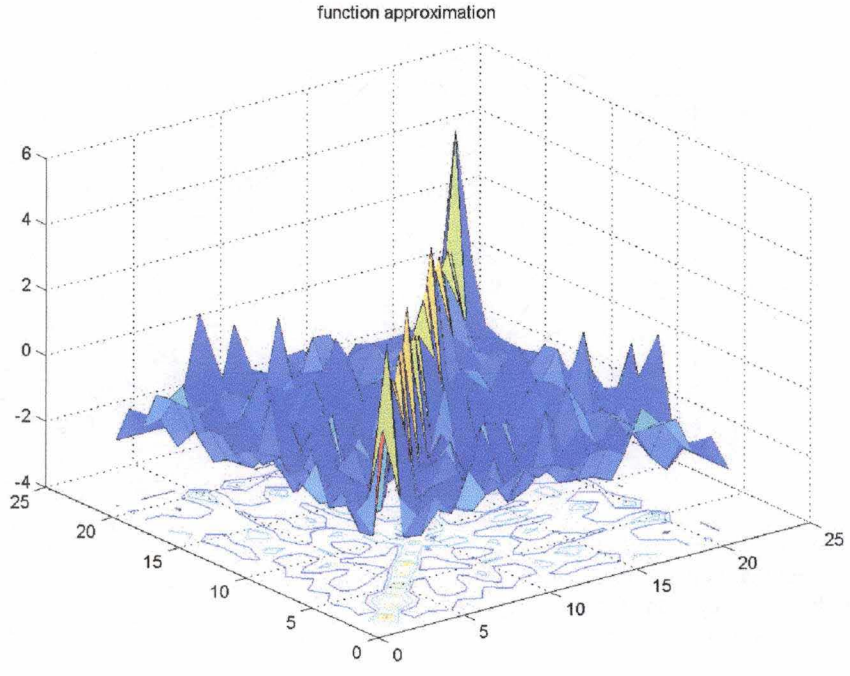


Figure 58 - Fonction crée par le réseau de neurones après apprentissage

Conclusion :

Les réseaux de neurones peuvent construire et reconnaître de manière précise une fonction discontinue présentant de fortes variations d'amplitude entre les points contigus. C'est le cas de la matrice de scores, et aussi des tableaux de probabilité que le réseau de neurones va devoir calculer.

7.3.2. Prototype développé avec une base de données de taille réduite

La structure d'un réseau de neurones destiné à l'analyse d'un problème particulier est critique, car elle conditionne aussi bien les performances de ce réseau que la validité des résultats obtenus. La mise au point de notre réseau nécessite de manipuler plus de 1 million d'informations, si on réalise les tests sur toutes les données disponibles dans la base de données HOMSTRAD. Ceci serait pénalisant au niveau de la charge et nécessiterait un temps considérable. Nous avons donc construit un modèle de test pour réduire les dimensions du système. Ce modèle reproduit le problème à traiter avec une similitude de 100%.

Nous avons construit une base d'exemple de protéines alignées avec un alphabet à cinq lettres. Cette base d'exemples comporte trente protéines (fictives) alignées, à partir desquelles notre modèle va être mis au point.

Liste des alignements des protéines modélisées :

Ils sont repris en annexe D.

Statistique de cette base d'exemples:

Dénombrement des paires de pseudo acides aminés:

1.283	A	B	C	D	E
A	98	47	11	13	22
B	48	126	10	8	56
C	15	39	137	51	79
D	50	55	40	101	10
E	12	29	32	86	108

Table 3 - Statistique de la base d'exemple

Probabilités d'apparition de chaque paire :

	A	B	C	D	E
A	7,64%	3,66%	0,86%	1,01%	1,71%
B	3,74%	9,82%	0,78%	0,62%	4,36%
C	1,17%	3,04%	10,68%	3,98%	6,16%
D	3,90%	4,29%	3,12%	7,87%	0,78%
E	0,94%	2,26%	2,49%	6,70%	8,42%

Table 4 - Probabilité d'appariement des acides aminés dans la base de test

Préparation des données :

La base d'exemples comporte uniquement des cas positifs. Plusieurs structures de réseaux de neurones ont été testées, et le modèle donnant le meilleur résultat est décrit ci-après. Il est constitué de deux valeurs d'entrée (les deux acides aminés formant la paire) et d'une seule sortie. La valeur à apprendre est « un » pour un cas positif et « zéro » pour un cas négatif.

Pour chaque association positive de la base d'exemples, on génère une valeur négative pour toutes les autres occurrences d'association. Les exemples ainsi construits sont ensuite entrés dans un générateur de nombres aléatoires, qui crée la répartition statistique qui va alimenter en entrée le réseau de neurones.

La structure retenue pour le réseau est la suivante :

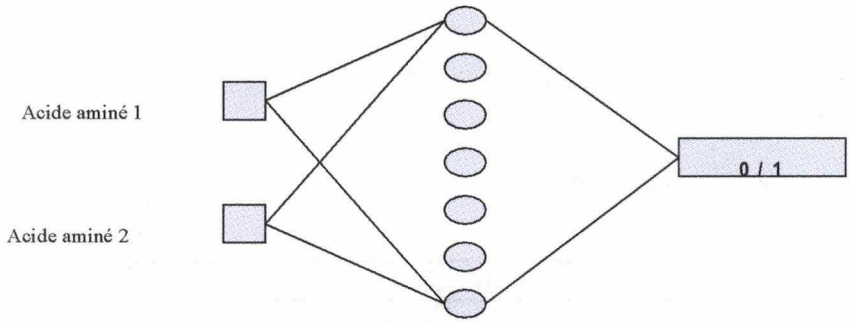


Figure 59 - Réseau de neurones pour le calcul de probabilités

Le réseau est entraîné avec la base d'exemples telle que décrite ci-dessus.

La fonction « objectif » que le réseau doit apprendre avec les exemples est une fonction dans R2, qui donne la probabilité de chaque occurrence d'alignement. Elle correspond à la table 3 ci-avant. Elle est représentée par la figure ci-après :

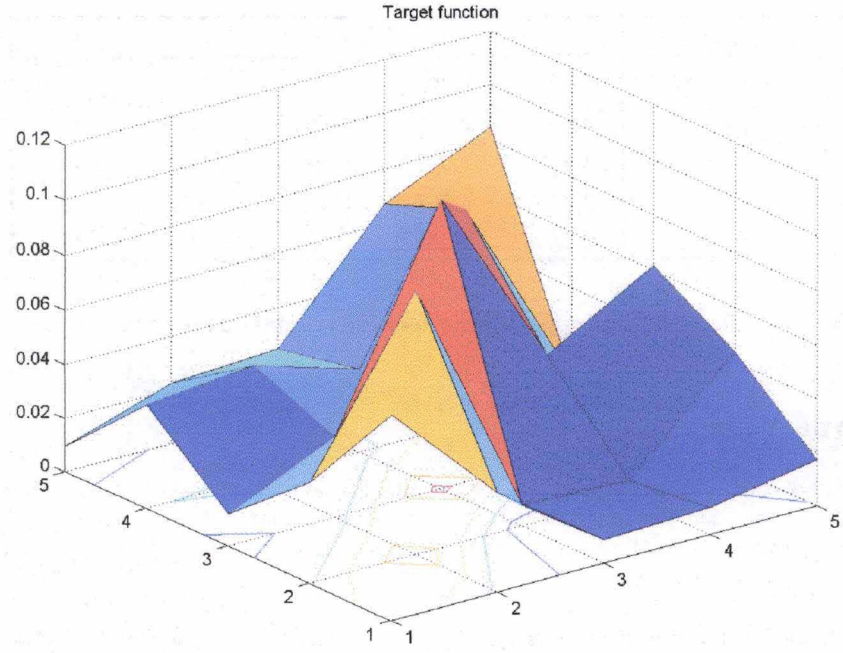


Figure 60 - Fonction de probabilité à découvrir par le réseau de neurones

Entraînement du réseau :

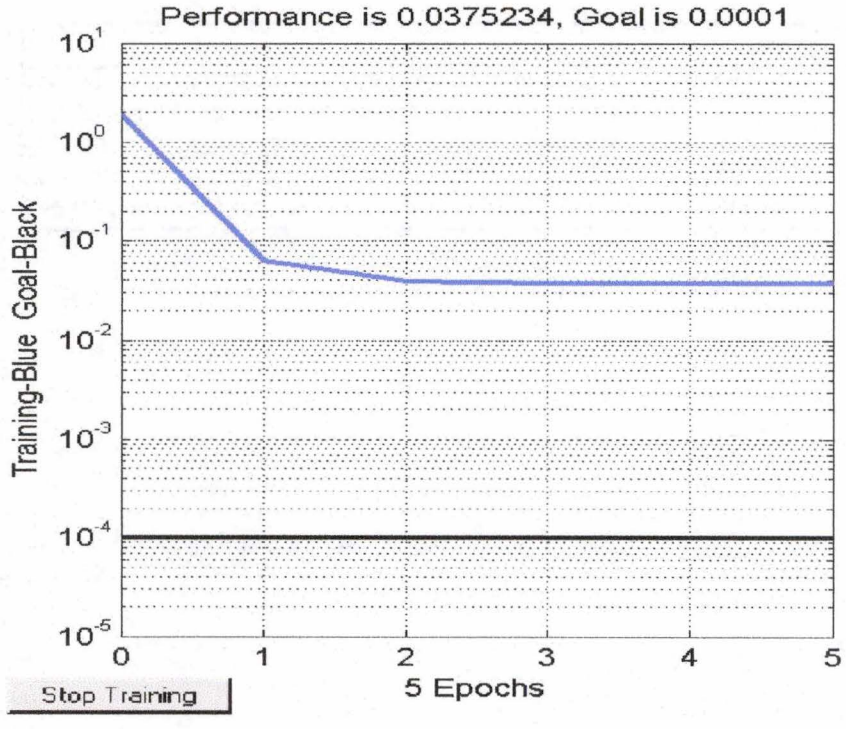


Figure 61 - Entraînement du réseau

Interrogation du réseau après entraînement :

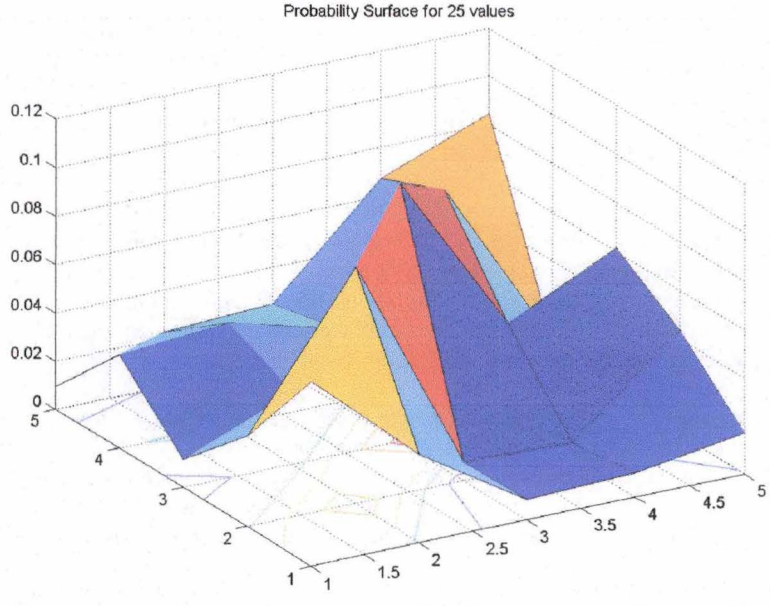


Figure 62 - Fonction discontinue de probabilités calculées par le réseau

Valeurs de probabilité calculées par le réseau :

	A	B	C	D	E
A	7,63830%	3,66330%	0,85736%	1,01330%	1,71470%
B	3,74120%	9,82070%	0,77942%	0,62354%	4,36480%
C	1,16910%	3,03970%	10,67800%	3,97510%	6,15740%
D	3,89710%	4,28680%	3,11770%	7,87220%	0,77942%
E	0,93531%	2,26030%	2,49420%	6,70300%	8,41780%

Table 5 - Probabilités calculées par le réseau de neurones

Résultat :

Il y a concordance entre le résultat obtenu, et la probabilité calculée à priori.
Il est donc possible de construire une matrice de scores avec ce modèle de réseau.

Il faut noter l'importance à accorder à la préparation des données d'apprentissage. La base de données d'exemples ne comporte que des cas positifs. Au cours du prétraitement, les cas négatifs sont générés et intégrés à la base d'exemples utilisée pour l'entraînement de réseau.

7.3.3. Application à la base de données HOMSTRAD

7.3.3.1. *Probabilités calculées à partir d'un échantillon*

La quantité d'information disponible dans la base de données HOMSTRAD est de 1,645 millions d'associations par paire d'acides aminés. Pour être représentatif, l'échantillon utilisé pour l'apprentissage du réseau de neurones sera également de taille significative, et de sa constitution dépendra la fidélité du résultat obtenu, en regard d'un traitement qui exploiterait de manière exhaustive l'entièreté de la base de données pour entraîner le réseau de neurones.

En effet, le réseau de neurone (utilisé dans le cadre de ce travail) permet de traiter des ensembles d'apprentissage dont la taille est inférieure à 100.000 informations. Nous avons donc fractionné le domaine d'étude et réalisé des associations de 7 acides aminés par rapport à 7 autres acides aminés, ce qui réduit la taille du domaine d'étude par un facteur 9. Nous avons donc le traitement en parallèle de 9 groupes d'association d'acides aminés de 7x7 positions en remplacement du traitement en globalité des 20 acides aminés (plus un gap), ce qui représentait 441 associations possibles.

Nous donnons en exemple les résultats obtenus par le réseau de neurones sur un sous ensemble de sept acides aminés, mis en correspondance avec sept autres acides aminés.
L'échantillon est représentatif de manière proportionnelle des associations considérées par rapport à leur présence dans la base de données. Le résultat obtenu par le réseau de neurones sera comparé à un calcul statistique réalisé en parallèle sur l'échantillon, afin d'évaluer l'erreur et de valider le fonctionnement du réseau.

Nous avons sélectionné pour ce test une zone quelconque associant sept acides aminés par rapport à sept autres. Le tableau ci-après définit la zone des associations {CDEFGHI} x {MNPQRST}.

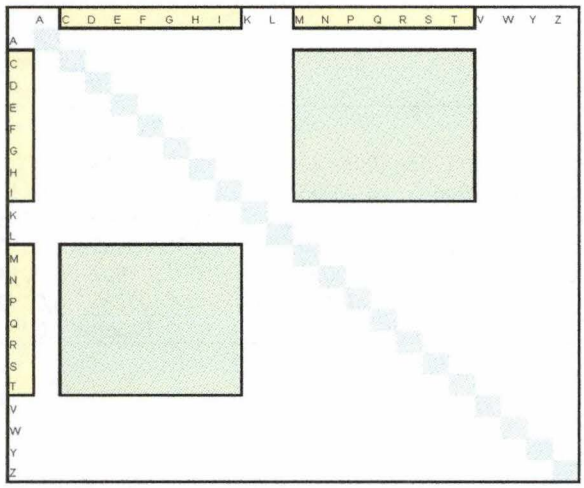


Figure 63 - Simplification et fractionnement du domaine d'étude

L'objectif étant de calculer des probabilités, la taille de l'échantillon d'un sous-système, rapportée à la taille totale des neuf sous-systèmes donnera sa représentativité par rapport au système global. Dans le test ci-après, l'échantillon étudié a un poids de 9,854%.

Les caractéristiques du réseau utilisé sont les suivantes :

```
% network specification
p = 2 ; % Number of inputs excluding bias
L = 49; % Number of hidden signals excluding bias
m = 1 ; % Number of outputs
MnMx = [1 7; 1 7] ;
  Nnet = newff(MnMx, [L, m] , {'tansig', 'purelin'}) ; % Network creation
  Nnet.trainParam.epochs = 5;
  Nnet.trainParam.goal = 0.00000001;
  Nnet = train(Nnet, COORD, REP) ;  % Training
```

Le réseau après entraînement est ensuite interrogé, et il restitue les probabilités suivantes :

Valeurs calculées par le réseau après entraînement

0.0034602	0.0051903	0.0077855	0.013841	0.0069204	0.0034602	0.026817
0.0043253	0.061419	0.035467	0.0095156	0.044983	0.017301	0.011246
0.0025952	0.025087	0.027682	0.0086505	0.025087	0.0069204	0.014706
0.0025952	0.027682	0.050173	0.0077855	0.019896	0.013841	0.011246
0.0034602	0.024221	0.037197	0.0095156	0.023356	0.014706	0.013841
0.0086505	0.052768	0.044983	0.012976	0.055363	0.012976	0.017301
0.0077855	0.035467	0.041522	0.013841	0.032007	0.011246	0.031142

La somme des probabilités calculées par le réseau de neurones sur un échantillon est égale à un, car le réseau de neurones que nous avons « construit » pour calculer des probabilités, restitue les probabilités relatives à l'ensemble des données qui ont été utilisées lors de l'apprentissage. Pour repositionner notre

sous-système dans le système global il faut multiplier les résultats calculés par le réseau de neurones par la représentativité du sous-système, soit 9,854 %. Nous obtenons alors la probabilité réelle des associations présentes dans ce sous système.

Ajustement proportionnel				représentativité	9,854%	
0,000340957	0,000511435	0,000767157	0,001363846	0,000681913	0,000340957	0,002642458
0,000426201	0,006052023	0,0034948	0,000937635	0,004432475	0,001704783	0,001108143
0,000255722	0,002471989	0,002727692	0,000852391	0,002471989	0,000681913	0,00144908
0,000255722	0,002727692	0,00494388	0,000767147	0,001960485	0,001363846	0,001108143
0,000340957	0,002386656	0,003665268	0,000937635	0,002301422	0,00144908	0,001363846
0,000852391	0,005199583	0,004432475	0,001278612	0,005455285	0,001278612	0,001704783
0,000767157	0,0034948	0,004091439	0,001363846	0,003153863	0,001108143	0,003068629

Le calcul statistique réalisé en parallèle, donne le résultat suivant.

Probabilité d'origine sur les 1.647.945 associations						
0,00037744	0,000507298	0,00076823	0,00134956	0,000731821	0,000350133	0,002603849
0,000404747	0,006039036	0,003524996	0,000951488	0,004432794	0,001679668	0,001112294
0,000285204	0,002452145	0,002720965	0,0008368	0,002474597	0,000720898	0,001440582
0,000288238	0,002733101	0,004920067	0,000800998	0,00197276	0,001348346	0,001128072
0,00035074	0,002418163	0,003631189	0,000930249	0,002286484	0,001429053	0,001381721
0,000827091	0,005152478	0,00442005	0,001286451	0,005409161	0,001296767	0,001682702
0,00077005	0,003492835	0,004097831	0,001412669	0,003125711	0,001103799	0,003075345

L'erreur résiduelle qui définit la précision du résultat obtenu avec un échantillon reflète dans le cas présent la « qualité » de bonne représentativité de cet échantillon par rapport à l'entièreté de la base de données.

L'erreur est de quelques pourcents, ainsi qu'indiqué ci-dessous :

9,67%	-0,82%	0,14%	-1,06%	6,82%	2,62%	-1,48%
-5,30%	-0,22%	0,86%	1,46%	0,01%	-1,50%	0,37%
10,34%	-0,81%	-0,25%	-1,86%	0,11%	5,41%	-0,59%
11,28%	0,20%	-0,48%	4,23%	0,62%	-1,15%	1,77%
2,79%	1,30%	-0,94%	-0,79%	-0,65%	-1,40%	1,29%
-3,06%	-0,91%	-0,28%	0,61%	-0,85%	1,40%	-1,31%
0,38%	-0,06%	0,16%	3,46%	-0,90%	-0,39%	0,22%

Cette erreur provient de la constitution de l'échantillon, et pas de la précision du calcul du réseau de neurones.
La fonction calculée est exactement la fonction attendue.

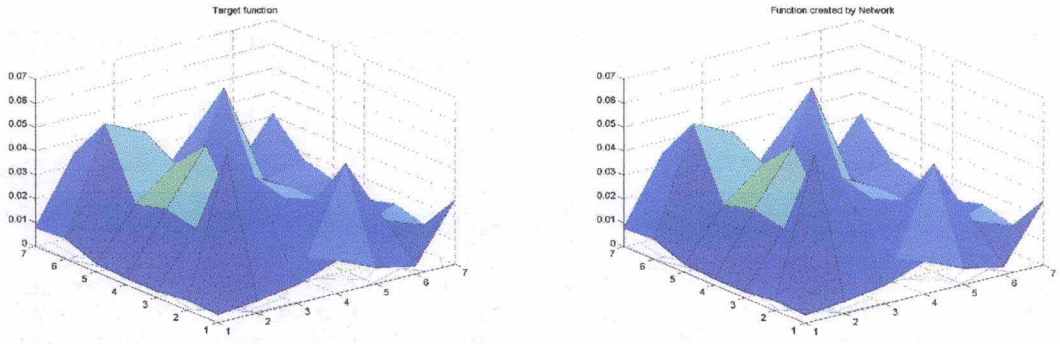


Figure 64 - Correspondance entre l'objectif et le résultat

L'entraînement pour arriver à un résultat stabilisé a été rapide :

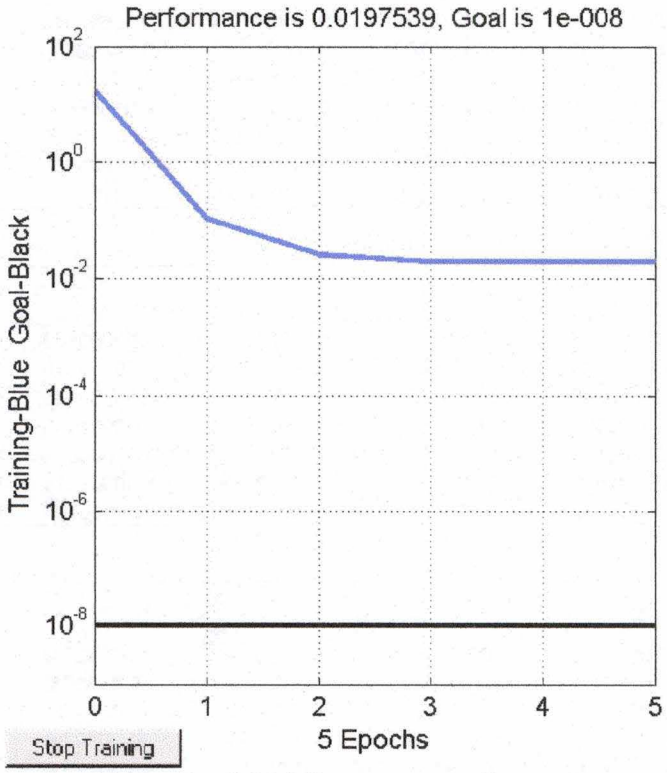


Figure 65 - Suivi de l'erreur en cours d'apprentissage

Les tableaux suivants mettent en évidence l'amplitude de l'erreur qui résulte du traitement par le réseau de neurones.

On constate que l'erreur est négligeable par rapport à celle qui est introduite par la constitution de l'échantillon qui est utilisé pour réaliser l'apprentissage du réseau.

Bien que chaque apprentissage utilise des valeurs initiales aléatoires pour les poids du réseau, l'erreur inhérente au réseau reste faible d'une exécution à l'autre, et négligeable par rapport à celle liée à la constitution de l'échantillon, ainsi que précisé ci avant.

Calcul à priori des valeurs objectif à atteindre sur base de l'échantillon

0,00346020761	0,00519031142	0,00778546713	0,01384083045	0,00692041522	0,00346020761	0,02681660900
0,00432525952	0,06141868512	0,03546712803	0,00951557093	0,04498269896	0,01730103806	0,01124567474
0,00259515571	0,02508650519	0,02768166090	0,00865051903	0,02508650519	0,00692041522	0,01470588235
0,00259515571	0,02768166090	0,05017301038	0,00778546713	0,01989619377	0,01384083045	0,01124567474
0,00346020761	0,02422145329	0,03719723183	0,00951557093	0,02335640138	0,01470588235	0,01384083045
0,00865051903	0,05276816609	0,04498269896	0,01297577855	0,05536332180	0,01297577855	0,01730103806
0,00778546713	0,03546712803	0,04152249135	0,01384083045	0,03200692042	0,01124567474	0,03114186851

Valeurs calculées par le réseau de neurones

0,00346020	0,00519030	0,00778550	0,01384100	0,00692040	0,00346020	0,02681700
0,00432530	0,06141900	0,03546700	0,00951560	0,04498300	0,01730100	0,01124600
0,00259520	0,02508700	0,02768200	0,00865050	0,02508700	0,00692040	0,01470600
0,00259520	0,02768200	0,05017300	0,00778540	0,01989600	0,01384100	0,01124600
0,00346020	0,02422100	0,03719700	0,00951560	0,02335600	0,01470600	0,01384100
0,00865050	0,05276800	0,04498300	0,01297600	0,05536300	0,01297600	0,01730100
0,00778550	0,03546700	0,04152200	0,01384100	0,03200700	0,01124600	0,03114200

Erreur due au réseau de neurones

0,0000000076	0,0000000114	-0,0000000329	-0,0000001696	0,0000000152	0,0000000076	-0,0000003910
-0,0000000405	-0,0000003149	0,0000001280	-0,0000000291	-0,0000003010	0,0000000381	-0,0000003253
-0,0000000443	-0,0000004948	-0,0000003391	0,0000000190	-0,0000004948	0,0000000152	-0,0000001176
-0,0000000443	-0,0000003391	0,0000000104	0,0000000671	0,0000001938	-0,0000001696	-0,0000003253
0,0000000076	0,0000004533	0,0000002318	-0,0000000291	0,0000004014	-0,0000001176	-0,0000001696
0,0000000190	0,0000001661	-0,0000003010	-0,0000002215	0,0000003218	-0,0000002215	0,0000000381
-0,0000000329	0,0000001280	0,0000004913	-0,0000001696	-0,0000000796	-0,0000003253	-0,0000001315

Erreur en pourcents

0,00022%	0,00022%	-0,00042%	-0,00122%	0,00022%	0,00022%	-0,00146%
-0,00094%	-0,00051%	0,00036%	-0,00031%	-0,00067%	0,00022%	-0,00289%
-0,00171%	-0,00197%	-0,00122%	0,00022%	-0,00197%	0,00022%	-0,00080%
-0,00171%	-0,00122%	0,00002%	0,00086%	0,00097%	-0,00122%	-0,00289%
0,00022%	0,00187%	0,00062%	-0,00031%	0,00172%	-0,00080%	-0,00122%
0,00022%	0,00031%	-0,00067%	-0,00171%	0,00058%	-0,00171%	0,00022%

7.3.3.2. *Calcul de scores d'affinité*

Matrice de scores de classes d'affinités physico-chimiques des acides aminés

Les acides aminés peuvent être regroupés en classes en fonction de leurs caractéristiques physico-chimiques.

Ces classes sont les suivantes :

- Aromatiques, non polaires : Phénylalanine (F), Tryptophane (W), Tyrosine (Y)
- Composé sulfuré, non polaire : Méthionine (M), Cystéine (C)
- Composé aliphatique, non polaire : Alanine (A), Isoleucine (I), Leucine (L), Valine (V), Proline (P)
- Petite dimension, neutre : Glycine (G)
- Groupement hydroxyle, polaire : Sérine (S), Thréonine (T)
- Groupement amide, polaire : Asparagine (N), Glutamine (Q)
- Chargé négativement, polaire : Acide Aspartique (D), Acid Glutamique (E)
- Chargé positivement, polaire : Histidine (H), Lysine (L), Arginine (R)
- On ajoutera la position « gap » qui résulte des alignements multiples présents dans la base de données HOMSTRAD.
-

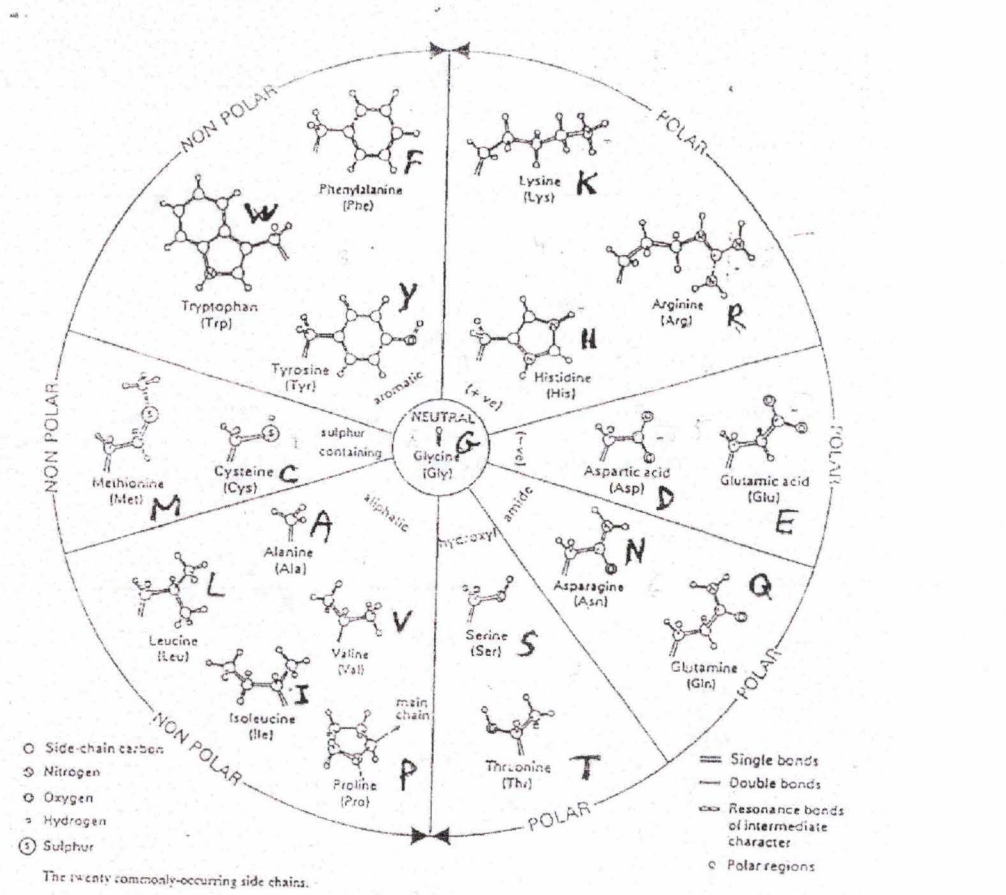


Figure 66 - Classes d'affinité des acides aminés

Constitution de l'échantillon :

L'échantillon construit pour l'entraînement du réseau va donc regrouper les informations disponibles dans la base de données, et réaliser des correspondances d'alignements d'acides aminés d'une classe, par rapport aux acides aminés d'une autre classe. Nous allons donc avoir neuf classes en correspondance l'une avec l'autre.

La matrice de score sera donc de 81 positions (9x9).

Les tableaux suivants donnent les résultats des probabilités calculés par le réseau de neurones, ainsi que les étapes de calculs pour construire la matrice de scores (voir les étapes du calcul au § 7.3.1.1).

Probabilités calculées par le réseau de neurones :

	AILPV	CM	DE	FWY	G	gap	HKR	NQ	ST
AILPV	0,156716	0,009013	0,013022	0,015671	0,00781	0,01924	0,015474	0,009652	0,019562
CM	0,008856	0,012424	0,00109	0,00200	0,00073	0,00200	0,00145	0,00109	0,00182
DE	0,012485	0,00091	0,039535	0,00273	0,00400	0,01018	0,009965	0,00854	0,00854
FWY	0,015353	0,00182	0,00273	0,038357	0,00127	0,00509	0,00436	0,00236	0,00327
G	0,00763	0,00073	0,00400	0,00145	0,032307	0,00618	0,00327	0,00309	0,00418
gap	0,019257	0,00200	0,010008	0,00527	0,00672	0,13357	0,009352	0,00672	0,009175
HKR	0,015194	0,00145	0,009489	0,00418	0,00327	0,00929	0,04142	0,00854	0,00836
NQ	0,009769	0,00109	0,008747	0,00236	0,00327	0,00691	0,00854	0,019085	0,00709
ST	0,019642	0,00182	0,00872	0,00363	0,00436	0,00939	0,00854	0,00672	0,035093

Regroupement triangulaire :

	AILPV	CM	DE	FWY	G	gap	HKR	NQ	ST
AILPV	0,156716	0	0	0	0	0	0	0	0
CM	0,017869	0,012424	0	0	0	0	0	0	0
DE	0,025506	0,00200	0,039535	0	0	0	0	0	0
FWY	0,031024	0,00382	0,00545	0,038357	0	0	0	0	0
G	0,015447	0,00145	0,00800	0,00273	0,032307	0	0	0	0
gap	0,038497	0,00400	0,02019	0,01036	0,01290	0,133568	0	0	0
HKR	0,030668	0,00291	0,01945	0,00854	0,00654	0,018642	0,04142	0	0
NQ	0,019421	0,00218	0,01729	0,00473	0,00636	0,01363	0,01708	0,019085	0
ST	0,039204	0,00363	0,01726	0,00691	0,00854	0,018561	0,01690	0,01381	0,035093

Calcul des P_{ij}:

	AILPV	CM	DE	FWY	G	gap	HKR	NQ	ST	
AILPV	0,156716	0	0	0	0	0	0	0	0	0,265534
CM	0,008935	0,012424	0	0	0	0	0	0	0	0,031353
DE	0,012753	0,00100	0,039535	0	0	0	0	0	0	0,09711
FWY	0,015512	0,00191	0,00273	0,038357	0	0	0	0	0	0,075131
G	0,007724	0,00073	0,00400	0,00136	0,032307	0	0	0	0	0,063291
gap	0,019249	0,00200	0,01009	0,00518	0,00645	0,133568	0	0	0	0,201957
HKR	0,015334	0,00145	0,00973	0,00427	0,00327	0,009321	0,04142	0	0	0,10179
NQ	0,00971	0,00109	0,00864	0,00236	0,00318	0,006815	0,00854	0,019085	0	0,066335
ST	0,019602	0,00182	0,00863	0,00345	0,00427	0,009281	0,00845	0,00691	0,035093	0,097505

0,265534	0,031353	0,09711	0,075131	0,063291	0,201957	0,10179	0,066335	0,097505

Calcul des coefficients « *Odd ratios* »:

	AILPV	CM	DE	FWY	G	gap	HKR	NQ	ST
AILPV	2,222655	0	0	0	0	0	0	0	0
CM	0,126718	12,63798	0	0	0	0	0	0	0
DE	1,531846	1,01676	4,192313	0	0	0	0	0	0
FWY	0,601567	0,62671	0,28907	6,795211	0	0	0	0	0
G	0,387149	0,30859	0,54797	0,24146	8,064971	0	0	0	0
gap	1,145331	1,00737	1,64239	1,08920	1,61050	3,274796	0	0	0
HKR	0,28594	0,22960	0,49599	0,28145	0,25591	0,228527	3,997648	0	0
NQ	0,359258	0,34165	0,87450	0,30892	0,49364	0,331505	0,82436	4,337279	0
ST	1,11286	0,87378	1,34004	0,69281	1,01718	0,692754	1,25151	1,56936	3,691216

Calcul des logarithmes :

	AILPV	CM	DE	FWY	G	gap	HKR	NQ	ST
AILPV	1,152284	0	0	0	0	0	0	0	0
CM	-2,980311	3,659694	0	0	0	0	0	0	0
DE	0,615272	0,02398	2,067746	0	0	0	0	0	0
FWY	-0,733202	-0,67412	-1,79053	2,764518	0	0	0	0	0
G	-1,36904	-1,69622	-0,86783	-2,05014	3,011669	0	0	0	0
gap	0,195765	0,01060	0,71580	0,12327	0,68751	1,711405	0	0	0
HKR	-1,806216	-2,12281	-1,01161	-1,82904	-1,96628	-2,129563	1,999151	0	0
NQ	-1,476909	-1,54939	-0,19347	-1,69467	-1,01846	-1,592899	-0,27865	2,11679	0
ST	0,154272	-0,19467	0,42228	-0,52947	0,02458	-0,529586	0,32367	0,65018	1,884096

Matrice des scores :

0,5	AILPV	CM	DE	FWY	G	gap	HKR	NQ	ST
AILPV	2	0	0	0	0	0	0	0	0
CM	-6	7	0	0	0	0	0	0	0
DE	1	0	4	0	0	0	0	0	0
FWY	-2	-2	-4	5	0	0	0	0	0
G	-3	-4	-2	-5	6	0	0	0	0
gap	0	0	1	0	1	3	0	0	0
HKR	-4	-5	-3	-4	-4	-5	3	0	0
NQ	-3	-4	-1	-4	-3	-4	-1	4	0
ST	0	-1	0	-2	0	-2	0	1	3

Validité du résultat calculé par le réseau de neurones :

Pour valider les résultats, la matrice des scores a aussi été calculée en réalisant un traitement statistique complet des associations de classes. Cette matrice théorique est comparée au résultat obtenu en faisant calculer les probabilités par le réseau de neurones. Il y a une forte correspondance, qui est due à la construction de l'échantillon, établie pour refléter une correspondance proportionnelle avec la base de données, ce qui permet de vérifier que les écarts du résultat obtenus sont dus essentiellement au soin apporté à la construction de l'échantillon, et que les erreurs dues au réseau de neurones proprement dit, tel qu'il est construit et utilisé, sont négligeables.

Théoriques	AILPV	CM	DE	FWY	G	gap	HKR	NQ	ST
AILPV	2	0	0	0	0	0	0	0	0
CM	-6	7	0	0	0	0	0	0	0
DE	1	-1	4	0	0	0	0	0	0
FWY	-2	-2	-4	5	0	0	0	0	0
G	-3	-4	-2	-4	6	0	0	0	0
gap	0	-1	1	0	1	3	0	0	0
HKR	-4	-5	-3	-4	-4	-5	3	0	0
NQ	-3	-4	-1	-4	-3	-4	-1	4	0
ST	0	-1	0	-2	0	-2	0	1	3

7.4. Discussion sur l'utilisation des réseaux de neurones pour le calcul de matrices de scores

Cette étude a permis de construire une nouvelle matrice de scores, qui exploite les données contenues dans la base de données HOMSTRAD. Par une approche statistique de calcul identique à celle à celle utilisée pour les matrices BLOSUM nous avons construit une matrice de score de référence, qui a été validée par un module de comparaison de performance du système MATCH-BOX.

Cette matrice utilisée comme « objectif à atteindre » par un réseau de neurones, a permis de montrer que le réseau de neurones construit et entraîné pour calculer des probabilités peut être utiliser pour construire des matrices de scores. La précision du résultat obtenu par le réseau de neurones est plus sensible à la nature de l'information contenue dans l'échantillon de données qui est utilisé pour l'entraînement du réseau, qu'à l'erreur inhérente au réseau lors de la restitution des résultats.

Le calcul de matrice de scores dynamiques par un réseau de neurones est une voie nouvelle. Elle demande cependant la manipulation d'une quantité importante d'informations, qui sont actuellement librement disponibles dans les bases de données mises à jour et enrichies en continu par la communauté scientifique.

Une contrainte de cette technique est l'obligation de manipuler de grandes masses de données qui peuvent constituer un facteur limitant, en regard des ressources informatiques dont on peut disposer. Mais, d'autre part elle épargne l'obligation de réaliser les comptages multiples nécessaires lors de l'étude statistique, ceux-ci étant fait de manière implicite et automatique par le réseau de neurones.

Un autre intérêt de cette approche est la possibilité de construire dynamiquement des matrices de scores, auxquelles on peut donner la généralité ou la spécificité qui correspond au domaine étudié, en réalisant une sélection appropriée des sous-ensembles de bases de données qui constitueront la base d'exemple utilisée pour l'apprentissage du réseau.

8. CONCLUSIONS

8.1. Sur le plan de la formation

Le cheminement réalisé au cours de ce travail de fin d'études nous a permis d'atteindre l'objectif principal fixé au départ, c'est-à-dire d'investiguer de manière approfondie le domaine de l'intelligence artificielle et de construire des repères qui vont nous permettre de maîtriser les relations que l'on pourra établir entre ces « outils » et les applications.

8.2. Sur le plan de l'intelligence artificielle

L'intelligence artificielle est entourée d'un certain mystère qui en fait un « monde à part » pour un non-spécialiste. L'approfondissement de nos connaissances réalisé dans le cadre de ce TFE, nous a permis de mieux cadrer les atouts et les faiblesses de cette discipline.
L'informatique traditionnelle cherche des solutions exactes à des problèmes donnés. L'intelligence artificielle propose une autre approche qui ne fournit pas une solution parfaitement exacte, mais qui permet de trouver celle qui s'en rapproche le plus.
Aujourd'hui, l'intelligence artificielle est présente dans de nombreux endroits, sans que l'on s'en rende compte. Son rôle est de plus en plus important dans l'automatisation de nombreuses tâches, dans les moteurs de recherche sur Internet, dans le contrôle de processus, dans la fouille de données. On tend vers une meilleure intégration de l'intelligence artificielle dans les applications informatiques « classiques ».
Nous sommes resté attentif et vigilent lors de son utilisation en évaluant constamment les risques et la qualité de l'information que peut fournir un tel système lorsqu'il est exploité, après apprentissage.
L'intelligence artificielle se révèle être un outil puissant dans des situations complexes où d'autres techniques sont inexploitables.

8.3. Sur le domaine d'étude

L'utilisation d'un réseau de neurones pour calculer une matrice de scores s'est révélée possible. Cette approche pourra permettre de construire de manière dynamique des matrices de score spécifiques d'un domaine d'étude déterminé. Le calcul de la matrice de scores par le réseau de neurones nécessite une préparation préalable des données extraites des banques de données biologiques. Pour permettre au réseau de neurones de calculer des probabilités, il faut pour chaque donnée positive extraite des banques de données, associer une donnée négative aux autres variables. Les réseaux de neurones trouvent des applications dans des domaines d'étude proches du domaine étudié :

- Classifier des protéines inconnues dans des familles.
- Identifier et classifier les protéines transmembranaires.

8.4. Sur les matrices de scores

Ce travail a permis de construire une nouvelle matrice de scores dont les performances ont été vérifiées pour l'utilisation du logiciel « MATCH-BOX ». Elle permet d'obtenir des résultats d'alignement de qualité équivalente à ceux obtenus avec les matrices « PAM » et « BLOSUM ».

9. ANNEXES

A. Annexe A - Théorie de l'apprentissage statistique

Introduction

Le but principal de la théorie de l'apprentissage statistique est l'étude du processus d'inférence inductive, c'est à dire enrichir ses connaissances, faire des prédictions ou prendre des décisions à partir de l'analyse d'un ensemble de données [229] [8] [145] [158] [163] [215].

Cela consiste à partir de l'observation d'un phénomène, à construire un modèle et à faire des prédictions à partir de ce modèle.

Pour atteindre cet objectif, l'hypothèse suivante doit être vérifiée : on suppose que les observations futures (les tests) sont bien en relation avec les données utilisées pour l'apprentissage. Toutes les informations sont échantillonnées de manière indépendante à partir de la même distribution (i.i.d. « *independant and identically distributed* »)

Formalisation

Soit un domaine d'observation $\mathcal{X}$ et un domaine de propriétés $\mathcal{Y}$. Les paires $(X,Y) \in (\mathcal{X},\mathcal{Y})$ sont des variables aléatoires distribuées selon une loi de probabilité P inconnue.

Soit une séquence de n paires (X_i, Y_i), échantillonnées (*i.i.d.*) de façon à respecter la distribution de probabilité P.

Le but est de construire une fonction $g : \mathcal{X} \rightarrow \mathcal{Y}$, qui prédit la valeur de Y à partir de X, soit g(X)=Y.

Le risque R(g) d'avoir une erreur $R(g) \neq Y$ devra être faible.

$$R(g) = P(g(X) \neq Y) = E[\mathbb{1}_{g(X) \neq Y}]$$

On démontre que par application de la formule de Bayes on peut construire un classifieur bayésien t(x) (pour X=x) qui définit le risque minimum de la famille de fonction g.

$$R(t) = \inf_{g} R(g)$$

R(t) est le risque de Bayes.

La fonction g = t est l'objectif à atteindre, mais comme la fonction de probabilité P est inconnue, on ne pourra jamais déterminer t, ni le risque R(t), directement à partir des données échantillonnées.

Une fonction candidate g pourra être sélectionnée à partir des données sur base d'un critère appelé « risque empirique ».

$$Rn(g) = \frac{1}{n}\sum_{i=1}^{n} \mathbb{1}_{g(Xi) \neq Yi)}$$

Lorsque le domaine d'observation $\mathcal{X}$ est infini, on peut minimiser $R_n(g)$ de façon à avoir une correspondance totale entre l'espace d'observation et l'espace d'hypothèse :

$$\forall X_i : g_n(X_i) = Y_i \quad \text{et} \quad R_n(g_n) = 0$$

On se trouve dans une situation où il y a surapprentissage pour la fonction g_n (« *overfitting* »). On a minimisé le risque empirique, mais pas le risque réel pour la fonction g_n

Minimisation du risque empirique

On travaille sur base d'un modèle $\mathcal{G}$ qui définit une famille de fonctions g, et on minimise le risque pour ce modèle :

$$g_n = \arg\min_{g \in \mathcal{G}} R_n(g)$$

Il n'est pas certain cependant que la fonction inconnue appartienne effectivement au modèle $\mathcal{G}$. C'est le risque structurel.

Minimisation du risque structurel

Soit une séquence de modèles { $\mathcal{G}_d$: d= 1,2, ... } de taille d croissante. Nous allons ajouter au calcul du risque empirique un facteur de pénalisation (ou de généralisation) qui est fonction de la taille (ou de la complexité) du modèle :

$$g_n = \arg\min_{g \in \mathcal{G}_d, d \in N} \left[R_n(g) + pen(d,n) \right]$$

Il faut rappeler que le choix de la classe $\mathcal{G}$ et du facteur de pénalisation nécessite une connaissance a priori du domaine concerné, et qu'il n'existe pas de règles pour définir le meilleur choix.

Limites supérieures du risque

Le risque lié à une fonction g_n dépend non seulement des données d'échantillonnage, mais il dépend aussi de la distribution de probabilité P qui est inconnue.

Le risque $R(g_n)$ lié à une fonction g_n construite par un algorithme d'apprentissage auquel on a soumis un échantillon de taille n est une fonction aléatoire, car elle dépend des données. L'estimation de $R(g_n)$ prend la forme d'une définition de limites probabilistes :

$$R(gn) = \mathbb{E}\left[1_{gn(X) \neq Y} \right]$$

Représentation graphique du risque

L'objectif dans la construction du modèle est la recherche de la meilleure fonction g. On essaye de minimiser $|R(g) - R_n(g)|$ et de faire tendre le risque vers $R(g^*)$ dans la classe de fonction $\mathcal{G}$. C'est le risque d'erreur d'estimation qui est lié à la sélection aléatoire des échantillons de données.

R^* = Risque de Bayes, le minimum du risque

$R(g^*) = \inf_{g \in \mathcal{G}} R(g)$

$g_n = \arg\min_{g \in \mathcal{G}} R_n(g)$

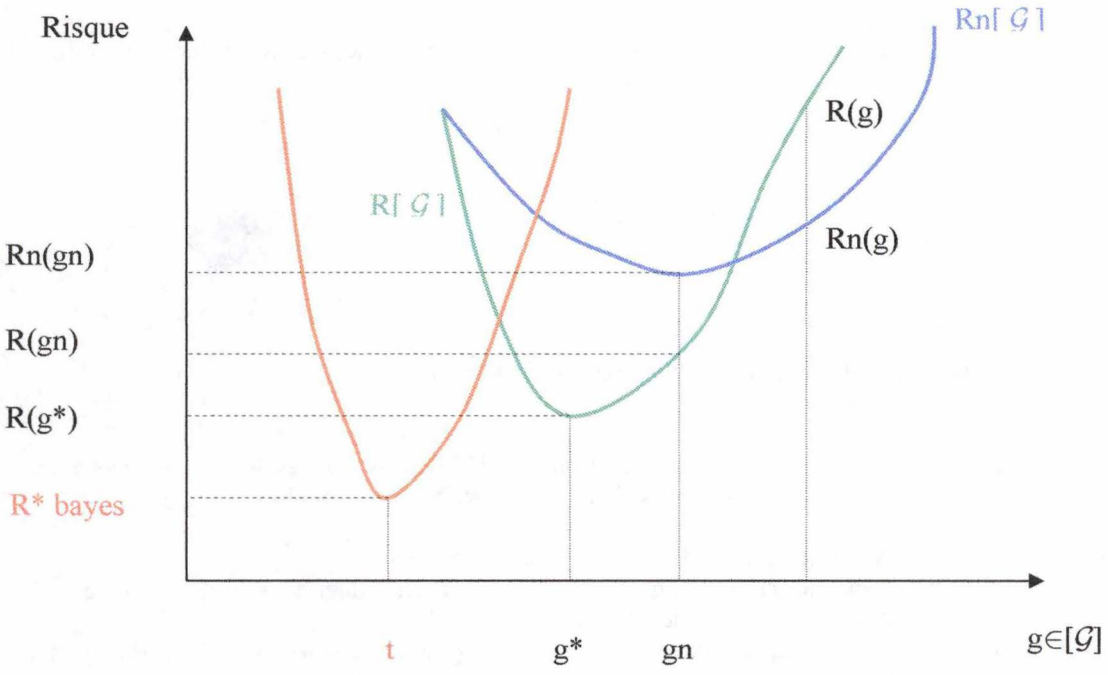


Fig A-1 : Graphique du risque

L'erreur d'approximation (Hoeffding) mesure la faculté des fonctions d'approcher le risque minimum objectif (risque de Bayes). Cette erreur est nulle si la fonction $t \in \mathcal{G}$.

L'erreur du modèle est $R(g_n) - R^*$. C'est l'erreur d'approximation, plus l'erreur d'estimation.

La borne supérieure de l'erreur, exprimée en fonction de la taille de l'échantillon n , du nombre de fonctions N dans la classe $\mathcal{G}$ et avec une probabilité au moins égale à $1 - \delta$ est de :

$$\sup_{g \in \mathcal{G}} (R(g) - R_n(g)) \leq \sqrt{\frac{\log N + \log 1/\delta}{2n}}$$

Théorème de Vapnik et Chervonenkis

Dans le cas où la classe $\mathcal{G}$ est non dénombrable, l'approche selon le calcul de Hoeffding n'est pas applicable ($N=\infty$). Dans ce cas, on va considérer une fonction de dimension liée à la taille de l'échantillon (Z1, ... Zn) et appelée fonction de croissance. Elle détermine le nombre maximum d'association de n points qui peuvent être classifiés par la classe de fonctions $\mathcal{G}$: $\mathcal{S}_{\mathcal{G}}(n)$.

Théorème de Vapnik et Chervonenkis : Pour tous $\delta > 0$ et avec une probabilité au moins égale à $1 - \delta$:

$$\forall g \in \mathcal{G}, \quad R(g) \leq R_n(g) + 2\sqrt{2\frac{\log \mathcal{S}_{\mathcal{G}}(2n) + \log(2/\delta)}{n}}$$

Dimension de Vapnik et Chervonenkis : la dimension VC de la classe $\mathcal{G}$ est le nombre n le plus élevé qui

vérifie

$$\mathcal{S}_{\mathcal{G}}(n) = 2^n$$

«PAC learning of an Artificial Neural Network»

L'apprentissage « Probablement Approximativement Correct » défini par Valiant est basé sur la théorie des probabilités. C'est une formalisation de la validité d'un apprentissage, qui a pour objectif :

- D'attirer l'attention sur le degré de doute ou de précision de celui-ci.
- De mettre en évidence les caractéristiques que doit avoir l'échantillon d'apprentissage, en ce qui concerne sa taille et ses caractéristiques statistiques.
- Pour caractériser parfaitement toute la connaissance du système, tous les exemples faisant partie de échantillon devront être iid (*« independent and identically distributed »*).

L'étude du système à construire pour analyser un problème devra comprendre deux volets :

- Quel est la taille et la nature de l'échantillon nécessaire pour caractériser au mieux la connaissance incluse dans le domaine étudié.
- Comment définir la structure de l'algorithme nécessaire et le nombre de ses paramètres.

Considérons un réseau de neurones qui comporte une seule valeur de sortie :

Soit :

Une fonction : $x \rightarrow t(x)$ avec $t(x) \in \{0,1\}$

L'ensemble des exemples : $X = \{0,1\}^n$ (n est le nombre de valeurs d'entrée du réseau de neurones).

Un échantillon de taille m : $S = ((x^1,t(x^1)), (x^2,t(x^2)),\ldots , (x^m,t(x^m)))$
Et $S(m,t)$ = l'ensemble de tous les exemples de dimension m, pour la fonction t.

Condition pour que le système soit « Approximativement Correct » :

Pour un algorithme L , $\exists$ h(x) : $L(s) \rightarrow h(x) = t(x)$ ou $L(s) \rightarrow h(x) \neq t(x)$

Soit :

t(x) la fonction à apprendre
h(x) la fonction découverte par l'algorithme
P = la probabilité d'erreur d'apprentissage
E_P = l'erreur d'apprentissage

Alors : $E_P = P(\{x \in X : h(x) \neq t(x)\})$

L'algorithme d'apprentissage est PAC si $\forall\ \delta, \varepsilon\ \exists$ un échantillon de taille $m_o(\delta, \varepsilon)$ tel que $\forall$ fonction t et $\forall$ distribution de probabilité P, il existe :

$m \geq m_o(\delta, \varepsilon) \Rightarrow P^m(\{s \in S(m,t) : E_P > \varepsilon\}) < \delta$

C'est-à-dire que si on fournit à l'algorithme un échantillon de taille au moins égale à $m_o(\delta, \varepsilon)$, alors il est probable qu'après apprentissage la fonction calculée par le réseau sera approximativement correcte.

L'algorithme est consistant PAC, si il n'existe qu'un nombre fini de fonctions.

Dimension de Vapnik Chervonenkis pour le réseau de neurones $\mathcal{N}$

Soit :

n le nombre d'entrées
1 la valeur de sortie $y \in \{0,1\}$
T les exemples présentés au réseau

Le réseau $\mathcal{N}$ atomise T (shatter), si pour chacune des $2^{(T)}$ possibilités de partitionner T en deux ensembles disjoints T^0 et T^1, il existe une fonction f calculable par $\mathcal{N}$ telle que :

$f(x) = 0$ si $x \in T^0$

$f(x) = 1$ si $x \in T^1$

La dimension de Vapnik Chervonenkis du réseau est définie par : $VC_{dim}(\mathcal{N})$ = taille maximum T telle que les exemples soient atomisés par $\mathcal{N}$.

Formulations théoriques

Théorème 1 : pour un réseau de neurones $\mathcal{P}^n$ avec n entrées et une seule sortie pouvant prendre la valeur 0 ou 1, la dimension de Vapnik Chervonenkis est donnée par :

$$VC_{dim}(\mathcal{P}^n) = n+1$$

Algorithme PAC : Si $\mathcal{N}$ calcule seulement un nombre fini de fonctions, alors cet algorithme est un algorithme PAC et la taille de l'échantillon $m_o(\delta , \varepsilon)$ peut toujours être déterminée. Il en résulte que pour tout algorithme PAC, ce qui est crucial, ce n'est pas la taille de l'échantillon, mais c'est la $VC_{dim}(\mathcal{N})$. Ce théorème a été établi par Blumer.

Théorème 2 : si un réseau de neurones a une dimension $VC_{dim} \geq 1$, alors tout algorithme d'apprentissage L consistant pour $\mathcal{N}$ est un algorithme PAC.

Les formules établies par ces théorèmes sont cependant difficilement calculables pour des réseaux de neurones ayant une structure plus complexe que celle définie par les hypothèses posées pour formuler ces théorèmes.

B. Annexe B – Rétro propagation

Formulation des équations du perceptron multi-couches [9]

Le perceptron multi-couches associe plusieurs neurones par couche et plusieurs couches successives. La sortie d'un neurone peut être transmise à l'entrée d'un ou de plusieurs neurones de la couche suivante. La figure ci-dessous représente un neurone, avec les notations utilisées dans les formules suivantes. Il s'agit du neurone *i* de la couche (*l*). On ajoute systématiquement un biais à l'entrée des neurones.

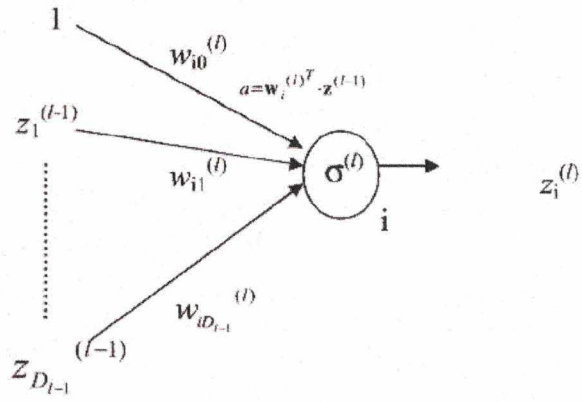


Fig. B-1 : neurone i de la couche (l)

Soit les notations suivantes pour la formalisation des entrées Z et des poids W :

$$\mathbf{z}^{(l)} = [1, z_1^{(l)}, \ldots, z_{Dl}^{(l)}]^T$$

$$\mathbf{W}^{(l)} = [\mathbf{w}_1^{(l)}, \mathbf{w}_2^{(l)}, \ldots, \mathbf{w}_{Dl}^{(l)}]^T$$

$$\mathbf{W} = \{\mathbf{W}^{(1)}, \ldots, \mathbf{W}^{(L)}\}$$

Ce qui permet d'écrire de manière synthétique l'expression analytique d'une couche :

$$\mathbf{Z}^{(l)} = [1, \sigma^{(l)}(\mathbf{W}^{(l)} \cdot \mathbf{z}^{(l-1)})^T]^T$$

L'application de cette formule permet de calculer la sortie du réseau. Pour L couches de poids, une entrée vectorielle **x** et une sortie scalaire y, on a :

$$\mathbf{z}^{(0)} = [1, x^T]^T$$
$$y = Z_1^{(L)}$$

[9] ELEC2870 – TP cours M. Verleysen

Rétro propagation du gradient

La phase d'apprentissage consiste à minimiser un critère d'erreur en adaptant l'ensemble des poids. Pour un problème de régression, il est habituel de choisir le critère des moindres carrés :

$$E = \sum_{p=1}^{P} E^p = \sum_{p=1}^{P} (y^p - t^p)$$

Pour chaque période p, on mesure l'erreur par rapport à la réponse objectif t.

Pour minimiser ce critère, nous utilisons la technique de descente du gradient. Le gradient stochastique par rapport à l'ensemble des couches de poids s'écrit :

$$\nabla_{\mathbf{W}} \mathbf{E}^p = 2.(y^p - t^p) \,.\, \nabla_{\mathbf{W}} y^p$$

Nous devons donc calculer le terme $\nabla_{\mathbf{W}} y^p$

Pour simplifier les notations, l'indice p est omis dans les formules suivantes.

Si on développe l'expression générale du gradient $\nabla_{\mathbf{W}} z_i^{(l)}$ d'un neurone i dans la couche (l), nous pouvons identifier trois composantes à ce gradient :

$$\nabla_{\mathbf{W}} z_i^{(l)} = \{ \nabla_{\mathbf{W}(k<l)} z_i^{(l)} , \nabla_{\mathbf{W}(k=l)} z_i^{(l)} , \nabla_{\mathbf{W}(k>l)} z_i^{(l)} \}$$

Où :

$$\nabla_{\mathbf{W}(k<l)} z_i^{(l)} = \{ \nabla_{\mathbf{W}(k=1)} z_i^{(l)} , \ldots , \nabla_{\mathbf{W}(k=l-1)} z_i^{(l)} \}$$

$$\nabla_{\mathbf{W}(k>l)} z_i^{(l)} = \{ \nabla_{\mathbf{W}(k=l+1)} z_i^{(l)} , \ldots , \nabla_{\mathbf{W}(k=L)} z_i^{(l)} \}$$

Etant donné que la sortie $z_i^{(l)}$ du neurone i de la couche (l) ne dépend pas des poids $\mathbf{W}_{(k>l)}$ des couches qui viennent après la couche (l), le gradient associé à ces poids ne contribue pas à $\nabla_{\mathbf{W}} z_i^{(l)}$:

$$\nabla_{\mathbf{W}(k>l)} z_i^{(l)} = \{ \mathbf{0} , \ldots , \mathbf{0} \}$$

Les deux autres contributions sont :

(1) $$\nabla_{w^{(l)}} z_i^{(l)} = \left. \frac{d\sigma^{(l)}}{da} \right|_{a = w_i^{(l)T} . Z^{(l-1)}} .0_i^{(l)} . Z^{(l-1)^T}$$

(2) $$\nabla_{w^{(k<l)}} z_i^{(l)} = \left. \frac{d\sigma^{(l)}}{da} \right|_{a = w_i^{(l)T} . Z^{(l-1)}} . \sum_{j=1}^{D_{l-1}} wij(l) . \nabla_{w^{(k<l)}} z_j^{(l-1)}$$

Ces formules de récurrence permettent de calculer successivement les contributions de toutes les couches de poids au gradient $\nabla_{\mathbf{W}} y$. En effet en commençant par la dernière couche (L), pour laquelle nous avons $y = Z^{(L)}$ et $\nabla_{\mathbf{W}} y = \nabla_{\mathbf{W}} z_i^{(L)}$, et en procédant de proche en proche en appliquant successivement l'équation (1) et (2), on peut remonter jusqu'à la première couche, d'où le nom de rétro-propagation.

Exemple en MATLAB

```
                % PARAMETRES

P=10000;
sig=0;
tot=1000;       % nombre d'époques d'apprentissage
d0 = 0;         % nombre d'entrées
d1 = 50;        % nombre de neurone de la couche cachée
d2 = 1;          % nombre de neurone de la couche de sortie
                % global WW1
                % global WW2
alpha1 = .001;  % convergence
alpha2 = .001;
b1=1;
b2=1;
slope = 0.5;    % pente th
                % global E;



[Z,tn] = gen_fun(P,sig);

nv  = size(Z,1);                % nombre de points d'entrée
d0 = nv ;
napp = size(Z,2);               % taille de l'échantillon d'apprentissage
w1=rand(d1 ,nv+1);
w2=rand(d2 ,d1+1);
                                %       cas de test paticulier
                                %       w1= 0.5 * ones(d1 ,nv+1);
                                %       w2= 1 * ones(d2 ,d1+1);
                                %
v =ones(1,nv);                  %       point qui est traité
N = zeros(1,tot);
O = diag(ones(1,d1)) ;          %       filtre colonnes
                                %
                                %       graphique à la demande
                                %       figure(1);  % ensemble d'apprentissage
                                %       j=1:1:napp;
                                %       plot3(Z(1,j),Z(2,j),tn);


%%%%%%%%%%%%%%%%%%%%%%%%%%  BOUCLE SUR PERIODES

for b=1:tot,

        E=zeros(1,napp);        %       plot erreur résiduelle
                                %       traite un point de l'échantillon

%%%%%%%%%%%%%%%%%%%%%%%%%%  BOUCLE SUR L'ECHANTILLON
```

```
for i=1:napp,
                          %        initialise un point
    for k=1:nv,
      v(1,k)=Z(k,i);
    end;
     tv=tn(i);


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%        couche d'entrée
   s0 = v ;
   z0 = [1 , v]';

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% couche cachée
   a1 = (w1 * z0) ;
   Point =  a1' ;
                       %[th,dth] = nl_fun(Point,slope) ;
                       %
    th = tanh(Point);
   dth = (1-(tanh(Point) .* tanh(Point)));
                       %
    s1 = th ;
    ds1 = dth ;
   z1 =  [1 , s1]' ;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%        couche de sortie
   a2 = (w2 * z1) ;
   fc2 = 1 ;
   z2 = fc2 * a2;
   y = z2;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%        calcul de l'erreur de sortie
   dErr2 = 2*(y-tv);
   E(1,i) =  dErr2/2;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%    formule 1  (nabla w2 z2)
   df2 = 1 ;
   NablaW2y= df2 .* z1';
   NablaE2= dErr2  .* NablaW2y ;          % erreur W2  %

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%        formule 1  (nabla w1 z1)
%    test – no more used
 % NablaW1y = zeros(d1,nv+1);
 % for j=1:d1,
 %   NaW2j = df2 .* ds1*O(:,j).*w2(:,j+1)'.*z0';
 %   NablaW1y(j,:) = NablaW1y(j,:)+ NaW2j;
 % end;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

   coefl = ds1' .* w2(:,2:d1+1)'  ;
   NablaW1y = df2  .*  (coefl * z0')  ;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

   NablaE1= dErr2  .* NablaW1y ;     %% erreur  W1


   w1 = w1 - alpha1 * NablaE1 ;
```

```
        w2 = w2 - alpha2 * NablaE2 ;

end;

%%%%%%%%%%%%%%%%%%%%% BOUCLE SUR L'ECHANTILLON

N(b) = E * E' ;
alpha1 = (b1*alpha1)/(b1+alpha1);
alpha2 = (b2*alpha2)/(b2+alpha2);

end;

%%%%%%%%%%%%%%%%%%%%% BOUCLE SUR PERIODES



%%%%%%%%%%%%%%%%%%%%% IMPRESSION DES FIGURES

for i=1:napp,

  xy1(i) = ((Z(1,i))*10)+100;
  xy2(i) = ((Z(2,i))*10)+100;

  c1=round(1+xy1(i)/2);
  c2=round(1+xy2(i)/2);

  XY1(c1)=Z(1,i);
  XY2(c2)=Z(2,i);
  XY3(c1,c2) = E(1,i);
  TY3(c1,c2) = tn(i);

end;

%%%%%%%%%

figure(2)
mesh(XY1,XY2,XY3);
figure(3)
mesh(XY1,XY2,TY3);
figure(1);
p=3:1:tot;
plot(p,N(1,p));

%%%%%%%%%

%%%%%%%%%%%%%%%%% CALCUL DE L'ECHANTILLON AVEC LES W1 ET W2

[xnv,xapp] = size(X);

for i=1:xapp,

        for k=1:xnv,
         xv(1,k)=X(k,i);
         end;
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%          couche d'entrée
        xs0 = xv ;
        xz0 = [1 , xv]';

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% couche cachée
        xa1 = (w1 * xz0) ;
        xPoint =  xa1' ;
                              %[xth,xdth] = nl_fun(xPoint,slope) ;
                              %
        xth = tanh(xPoint);
        xdth = (1-(tanh(xPoint) .* tanh(xPoint)));
                              %
        xs1 = xth ;
        xds1 = xdth ;
        xz1 =  [1 , xs1]' ;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%couche de sortie
        xa2 = (w2 * xz1) ;
        xf2 = 1 ;
        xz2 = xf2 * xa2;
        R(i) = xz2;
end;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% IMPRESSION DES FIGURES


figure(4);
 g=1:1:xapp;
 plot3(X(1,g),X(2,g),R);

%test
W1 = w1 ;
W2 = w2 ;
[r,c] = size(X);
x = ones(r+1,c);
x(2:r+1,:) = X;
s1 =ones(d1,c);
s1(2:d1+1,:) = tanh(W1 * x);
R = W2 * tanh(s1);

figure(5);
plot3(x(2,:), x(3,:), R);
```

C. Annexe C – Formation à l'outil « réseau de neurones»

Dans la première étape de notre étude, nous avons réalisé la programmation complète (ab initio) d'un réseau de neurones à deux couches (*feed forward neural network*) par calcul matriciel, y compris la rétro propagation du gradient.

Programmation d'un réseau de neurones à deux couches

Nous avons paramétré ce réseau de neurones pour faire varier le nombre de neurones dans chaque couche, ainsi que le nombre de périodes d'apprentissage.

Le réseau est testé sur une fonction continue définie dans R^2. La couche d'entrée comporte deux neurones pour enregistrer les coordonnées du point dans R^2. La couche cachée aura un nombre de neurones qui sera variable en fonction du test réalisé. La fonction de transfert est une tangente hyperbolique. La couche de sortie comprend un neurone qui enregistre ou qui restitue la valeur de la fonction continue dans R^2.

La structure du programme est donnée en annexe B.

Fonctions étudiées

$$f : R2 \rightarrow R : f(x) = RC(|x1|) + \sin \|x\| \ / \ \|x\|$$
$$f : R2 \rightarrow R : f(x) = [\exp(-\|x\| / a) - c^* \exp(-\|x\| / b)] \ / \ (1 - c)$$

La représentation en trois dimensions de l'échantillon des données d'apprentissage, donne l'allure de la fonction.

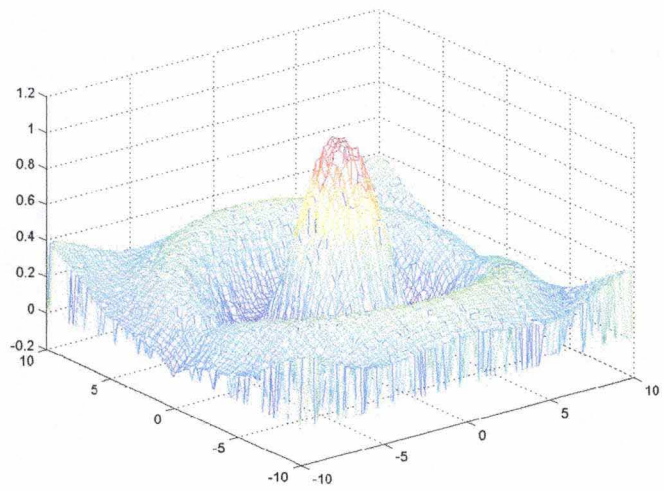


Fig. C-1 : Représentation 3D d'une fonction continue

Apprentissage

On génère des points représentatifs des fonctions ci-dessus pour constituer un échantillon d'apprentissage de 10400 points. Le test est réalisé avec 500 neurones dans la couche cachée et l'apprentissage dure 1000 périodes.

Le dimensionnement du réseau pour ce test est réalisé de manière empirique, en faisant croître progressivement le nombre de neurones de la couche cachée, et le nombre de périodes d'apprentissage.

Le nombre croissant de neurones cachés entraîne une diminution de l'erreur résiduelle (figure C-3), et le carré de l'erreur moyenne (MSE) diminue en fonction du nombre d'époques d'apprentissage.

La descente du gradient est continue, mais relativement lente au cours des 1000 périodes d'apprentissage.

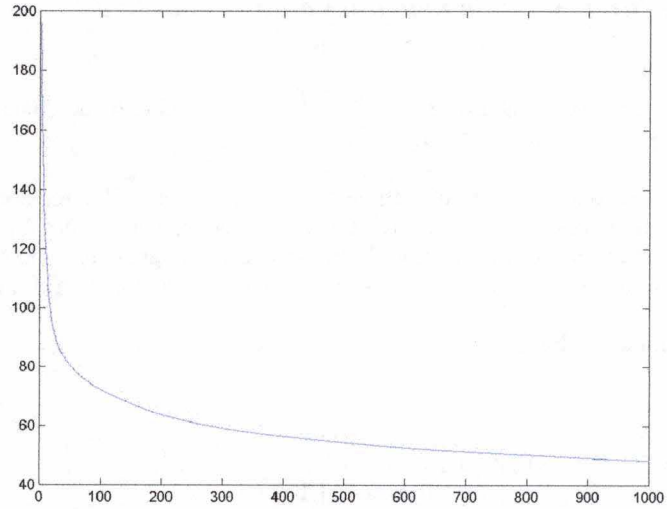


Fig. C-2 : Evolution de l'erreur (MSE) en cours d'apprentissage

Le graphique suivant donne la surface d'erreur résiduelle en fin d'apprentissage.

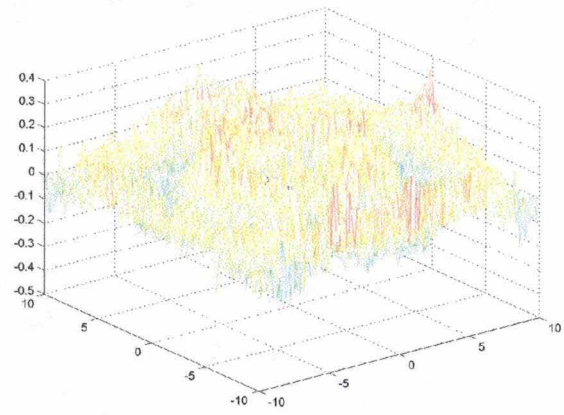


Fig. C-3 : Erreur résiduelle en fin d'apprentissage

Interrogation du réseau

On constitue une grille de points dans l'espace de variation de la fonction. On applique à ces points les poids calculés lors de l'apprentissage et on obtient la réponse donnée par le réseau. La représentation dans l'espace de la réponse du réseau redonne la fonction, avec la précision correspondant à la qualité de l'apprentissage du réseau.

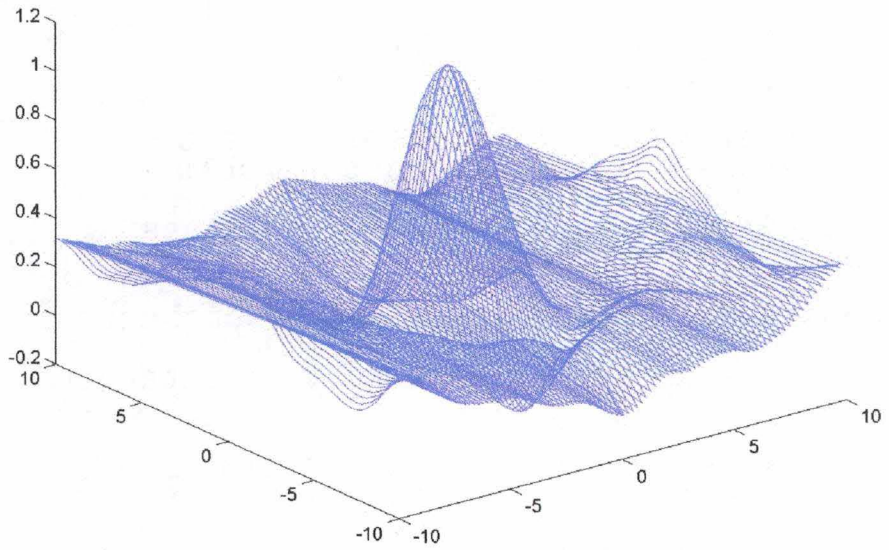


Fig. C-4 : Fonction 3D restituée par le réseau de neurones

On constate que la fonction est bien approximée, mais n'est pas reproduite parfaitement. Cela s'explique par le fait que l'erreur résiduelle en fin d'apprentissage reste importante.

Cette première partie de l'étude nous a permis de construire, de manipuler et de paramétrer des réseaux de neurones. Elle est réalisée par programmation directe des opérations de calcul matriciel de descente et de rétro propagation du gradient (voir annexe B).

D. Annexe D – Base de données de taille réduite

Liste des alignements des protéines modélisées :

CEBBCDBEEBDDBEBEBBDECDDDBDCCADCBBCBACEBCEDADADB
EDBACCBBDBBADEEEBBAECBAABDEDACCBBDBDBEADDDAABBB

ACBDBDBEACEDCCCAACBCDDDBEBECDBBAEDDBEECCDCAEACCCECBE
AEBAEBBDBDDBCECCBEBEBDAEEBDCDBBEDBCBDDCEDBBDEBCEBEBE

DACACDEEAACEDBCEEBEDBCCEEEBCEACCB
DBEAEECEBAEEDBCDCADDABBDAEBDDABEB

CEEDCEDECDCEBDEDDDDBBEECAADDAEEBCDEADEADD
EECDEDBDEDBBAAEDCCABADEAAADDDDCCCDDABDADA

CEAACCECEACDACBACBBBBABBADBDBDBBEBCBACBDBB
EEAACEEDCABCCCABEBBEBBEEACBBBDEBEBEEBCABBB

BCCAEEBDDBABCECBAEEAECEDBEEAAAADEDDEBCECDCCDCBECEBDCAC
B
BDAADAEDDBAECECBECEDEDEDEDDAAAACBBBDADDBBEDBCCCCEEBCAC
B

EEDADDEBBACADDBCAADCDABBDCDEDDDDBDEDBAC
BEDDAAEBEBCACDDADADCCAEEADAEDDBEEBAAEAE

EEEDACEBACBDCCBCEABEBCECDBADBBEAEECABDBCCDCDEE
BBADADDBADBDCCBDCABABDEDDBADAABBCDCBADDEDBCBED

CDDBBEEBBCDBEBCCDBBDBDCACBDEEECECEBBECDCDDCBCDEECCCAC
BDCDADEBBEABCCCEAEEDCDEACBABCDBEDDBAEEACDDEEDADEBCCBD

CBACCCCEEBADCDEDABCDACBACCCBCEDCDCA
CBEECDCEEEECACACBBCCACAABDDACDDCBBB

EDEEAEBCCCBDCDDBCBCEECBCCDDEBDBBCCBEBCEBBADDDDCACBABCE
D
EBAEAECCBCAAADABABEADCDCCEAEBAEBDEBCACEBBAEBDACACAAEEDD

CEBBDBCDBAABEBCDDECCCBCCEBBECEAA
EEABEACDBAAEBBDCDEECEABDDBEECEAA

BCADDDBBAEDDECCDBCACEAAADBDBEABBDBEAAACCDDADCCCDCBC
DCADBBEBBDDACBEDBBADEBABABABEEABBEDABACCDDBACCEDDBC

BADADACCBDDEAAEDEDAEBABAECDEEADBBCDBDCCCDAAE
BACACADCBCDEBADCDBADBABAEABDDADBBDAADCCEBEAD

BDECCEDBCECEECDEECBCBCCBBDECDECCDCDCAACAADCEEEBBCECCBE
B
BCDCBEDECDCDDCBDDCBDECBEADBCBEDCDCBDAADEBBDEBEBEEEEEAE
E

AECAEEBACCCCAEBBBECAACADEDCCDBBBECADDDCEEBDCEDBCAAABAB
DEECAACCAACEBEBEED
CCCAEDEDACDCAABBEECAEEBBEDCDDBCBECCADDCEDBDCEBCCEDDBBB
BEDAAADCBEAEEEEEEC

CBBACECEEBEDBECAACACBEDCDEBAABCDBCCACCDEACE
CEACEEDECBEDADCBABAEECDBDDAAAAEDBECCBEDEDCE

CEEBCCCEDCECECBBAEBDCEDC
EEDBCCEDAECBDCBAADADECEE

EACDEBEDEDBECAAAECBEBADCCABCCAA
EABCCBDDADEECAEADBBEBEDCEAEEBAA

CCBDCEBAEACDCBDDADEECACBBEBCBDADDBCCCDECB
EDAAEDAABBCECBCCEDDEEADBBCBCBAABDBCDCDEEE

ACBDDEEDAAAAEECDDEAAB
BCADDEDCBABADDCDBDACE

EEEDBDAEABCCCDEDBDEDAECCDEBDAAEECAEEECBDCBC
DCEBBDEEEACEEEEBEDDDAECBDCEDBBEBCAEEECBCCBD

DEEDAECECEBABBCCAABAEBAEEDBBBBECACCECDDCABDCAA
EBBCBBADCEEDBEECAABBEBBDCCAABEEEAEECACDDEBDDAE

DCAACEEEDCECACCCEEDBDACDBBBBCBCADEECEDCEAEEDEECEDBBAEB
DCECE
DEBACBECDCEBACECCBCBDECDBBBBEEEADEEEDACEADADBECEDBEABE
ACEBC

ABBBADAAEDEBBCDEBECDBBCDCEABDDBE
AAABBABBEADEEEAEBDCCAACBCEBEEDBB

ABCBDCCDBDDBEBCECEADDCCEADADDEBCCDCBDCEDEBBDBBCCDEEDCB
CBBADCCBBBCADBDDBEBCBCEDEAADAECCCBBEDCEBDDADBBBBCDBCEB

EABDBCDDECEDDDBCDCEACEECEC
BBBBDDABECDBAAACACCCECBDCC

CCDECEECAACCDBCAEEDEBBDDCACEC
CACCCDEEDADCAAEAADDDBBBBCCCDC

DDDBCCDDECADCCBAACACACD
DBBBCEDDBBBDDCCBCCECACD

BBBDDACAEEBDACECDBCEDCAADEADDEBBDDDADACCBCCCDCAEEEEEAC
CECEC
BABDDDAADEEDACEDACEEACEAABDCDEEBACBACBABBCCCBEBCEEBBBE
CEBEB

E. Annexe E – Modélisation du prototype de réseau de neurones

Etude de diverses structures de réseaux de neurones

Pour représenter le système étudié, plusieurs structures de réseaux ont été étudiées. Nous allons les décrire brièvement. Dans tous les modèles étudiés, nous n'avons considéré qu'une seule couche cachée :

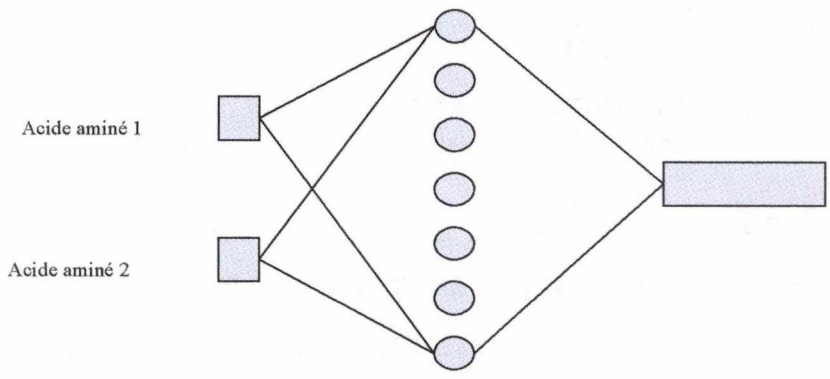
- Réseau type 1 : les acides aminés sont représentés en entrée par un vecteur binaire. La couche cachée comporte de deux à cinquante neurones. La couche de sortie comporte autant de sorties que de réponses souhaitées. C'est une construction que l'on retrouve pour les recherches d'alignement de structure, mais ce type de réseau n'a pas donné de résultats exploitables pour l'analyse des alignements de séquences.

- Réseau type 2 : les acides aminés sont représentés en entrée par les coordonnées de la table de scores et les valeurs de la table de scores ou les probabilités associées sont chacune représentées par un neurone de sortie. Pour la table complète, cela représente 441 valeurs de sortie. Le nombre de neurones cachés varie de 21 à 441. Cette structure s'est révélée trop complexe, et nous n'avons pas obtenu de convergence.

- Réseau type 3 : les acides aminés sont représentés en entrée par les coordonnées de la table de scores et les valeurs de la table de scores ou les probabilités associées sont chacune encodées dans le même neurone de sortie. Le nombre de neurones cachés a été étudié dans une plage qui varie de 100 à 1000. C'est la structure qui a été retenue, et dont le dimensionnement est examiné plus en détail ci-après.

Etude du dimensionnement du réseau de neurones de type 3

Schéma du réseau

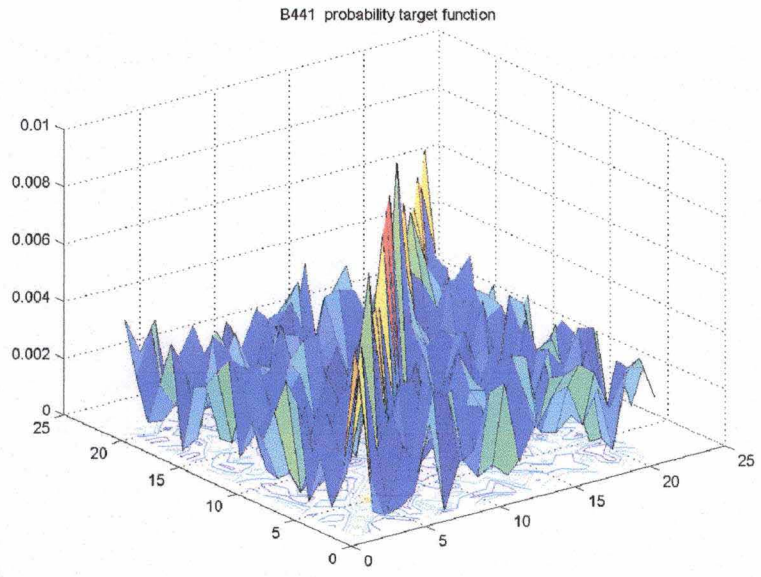


Fonction de probabilités apprise par le réseau

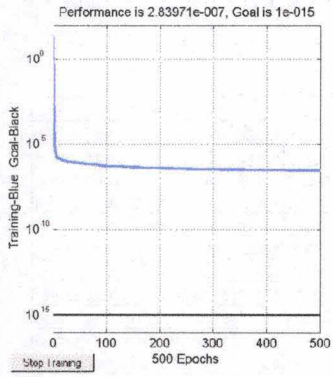
Le dimensionnement du réseau est réalisé avec les données d'entrée suivantes. C'est une table de 441 positions, donnant des probabilités d'association entre deux éléments de la table. C'est une table de pourcentages. La somme de toutes les cellules est égale à 1.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
1	0,00665	0,00208	0,00049	0,00060	0,00100	0,00322	0,00011	0,00163	0,00193	0,00380	0,00399	0,00112	0,00212	0,00375	0,00368	0,00394	0,00350	0,00210	0,00319	0,00361	0,00210
2	0,00212	0,00555	0,00044	0,00035	0,00247	0,00137	0,00406	0,00376	0,00137	0,00067	0,00093	0,00044	0,00334	0,00182	0,00280	0,00299	0,00149	0,00261	0,00033	0,00308	0,00296
3	0,00068	0,00173	0,00601	0,00224	0,00345	0,00067	0,00427	0,00212	0,00369	0,00296	0,00280	0,00091	0,00431	0,00413	0,00133	0,00117	0,00163	0,00378	0,00056	0,00315	0,00142
4	0,00222	0,00243	0,00177	0,00443	0,00046	0,00165	0,00061	0,00242	0,00214	0,00145	0,00317	0,00303	0,00397	0,00131	0,00156	0,00347	0,00119	0,00210	0,00025	0,00065	0,00254
5	0,00054	0,00128	0,00140	0,00376	0,00793	0,00075	0,00376	0,00375	0,00144	0,00263	0,00432	0,00320	0,00124	0,00322	0,00268	0,00196	0,00361	0,00280	0,00404	0,00392	0,00086
6	0,00373	0,00184	0,00308	0,00123	0,00380	0,00615	0,00200	0,00331	0,00068	0,00091	0,00018	0,00371	0,00224	0,00392	0,00217	0,00042	0,00338	0,00362	0,00128	0,00067	0,00345
7	0,00436	0,00032	0,00100	0,00215	0,00068	0,00303	0,00609	0,00105	0,00250	0,00019	0,00203	0,00117	0,00026	0,00081	0,00119	0,00147	0,00424	0,00032	0,00382	0,00245	0,00112
8	0,00305	0,00082	0,00299	0,00011	0,00046	0,00105	0,00285	0,00578	0,00212	0,00058	0,00193	0,00040	0,00331	0,00361	0,00200	0,00077	0,00313	0,00261	0,00236	0,00100	0,00310
9	0,00191	0,00044	0,00128	0,00116	0,00030	0,00264	0,00303	0,00361	0,00800	0,00205	0,00004	0,00079	0,00060	0,00100	0,00271	0,00315	0,00091	0,00096	0,00231	0,00275	0,00331
10	0,00236	0,00054	0,00168	0,00102	0,00142	0,00327	0,00117	0,00016	0,00284	0,00854	0,00142	0,00294	0,00035	0,00051	0,00140	0,00053	0,00130	0,00077	0,00301	0,00056	0,00068
11	0,00095	0,00158	0,00228	0,00243	0,00331	0,00152	0,00161	0,00172	0,00396	0,00079	0,00872	0,00158	0,00175	0,00105	0,00194	0,00327	0,00172	0,00011	0,00410	0,00366	0,00063
12	0,00415	0,00287	0,00387	0,00072	0,00354	0,00417	0,00422	0,00000	0,00243	0,00326	0,00313	0,00438	0,00081	0,00257	0,00047	0,00287	0,00200	0,00389	0,00137	0,00042	0,00364
13	0,00355	0,00093	0,00124	0,00284	0,00117	0,00186	0,00084	0,00215	0,00334	0,00049	0,00133	0,00046	0,00919	0,00327	0,00250	0,00296	0,00263	0,00047	0,00291	0,00324	0,00187
14	0,00289	0,00088	0,00329	0,00301	0,00086	0,00436	0,00242	0,00114	0,00138	0,00238	0,00025	0,00298	0,00200	0,00634	0,00336	0,00242	0,00168	0,00135	0,00378	0,00338	0,00039
15	0,00264	0,00261	0,00107	0,00270	0,00226	0,00247	0,00368	0,00350	0,00044	0,00000	0,00019	0,00193	0,00303	0,00044	0,00711	0,00026	0,00107	0,00417	0,00313	0,00236	0,00343
16	0,00025	0,00051	0,00401	0,00341	0,00103	0,00259	0,00229	0,00007	0,00060	0,00170	0,00196	0,00154	0,00217	0,00067	0,00037	0,00450	0,00396	0,00287	0,00177	0,00096	0,00088
17	0,00382	0,00263	0,00061	0,00383	0,00273	0,00222	0,00004	0,00348	0,00418	0,00280	0,00215	0,00254	0,00079	0,00310	0,00219	0,00285	0,00615	0,00429	0,00333	0,00343	0,00411
18	0,00219	0,00131	0,00137	0,00284	0,00207	0,00333	0,00203	0,00033	0,00149	0,00296	0,00054	0,00373	0,00387	0,00371	0,00096	0,00226	0,00028	0,00599	0,00268	0,00035	0,00039
19	0,00068	0,00060	0,00266	0,00368	0,00070	0,00051	0,00011	0,00128	0,00320	0,00336	0,00422	0,00173	0,00128	0,00310	0,00261	0,00184	0,00170	0,00047	0,00672	0,00110	0,00329
20	0,00207	0,00364	0,00047	0,00175	0,00030	0,00035	0,00320	0,00214	0,00338	0,00275	0,00338	0,00112	0,00154	0,00320	0,00404	0,00040	0,00263	0,00040	0,00151	0,00609	0,00347
21	0,00392	0,00266	0,00364	0,00058	0,00040	0,00147	0,00152	0,00273	0,00137	0,00112	0,00168	0,00032	0,00422	0,00049	0,00100	0,00154	0,00142	0,00294	0,00149	0,00012	0,00707

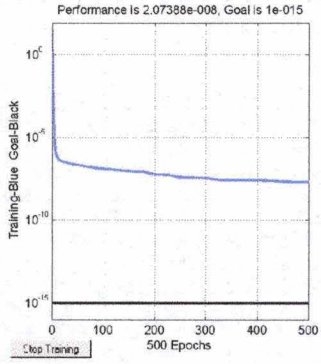
La table représentée sous forme graphique est un ensemble de points discontinus, dont les valeurs sont très variables entre une cellule quelconque et toutes les autres cellules adjacentes. La configuration de cette table est similaire à une matrice de scores, mais les valeurs qu'elle contient sont fictives. Cette table est représentée dans l'espace, dans la figure ci-dessous.



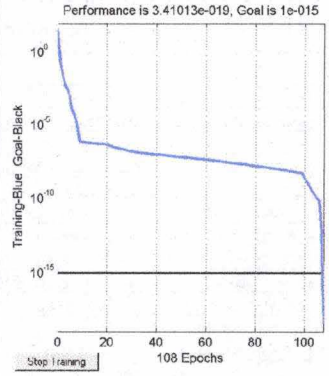
Etude de la convergence en fonction du nombre de neurones de la couche cachée



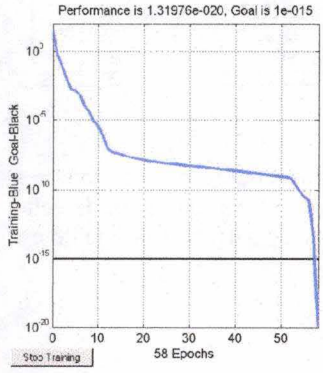
100 neurones cachés



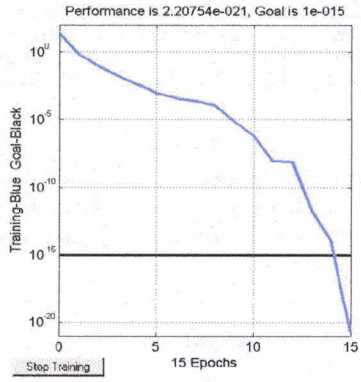
150 neurones cachés



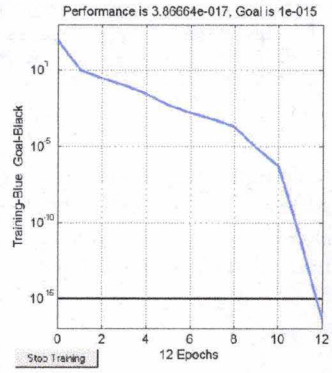
200 neurones cachés



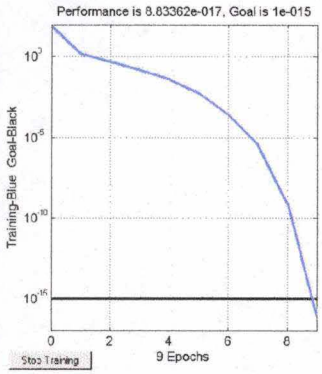
225 neurones cachés



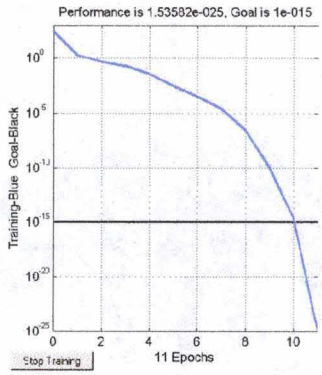
250 neurones cachés



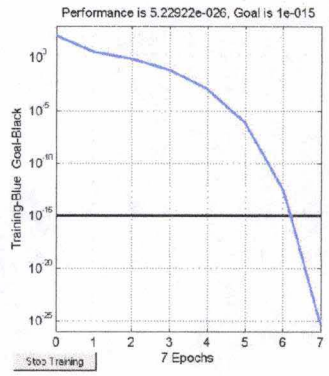
350 neurones cachés



400 neurones cachés

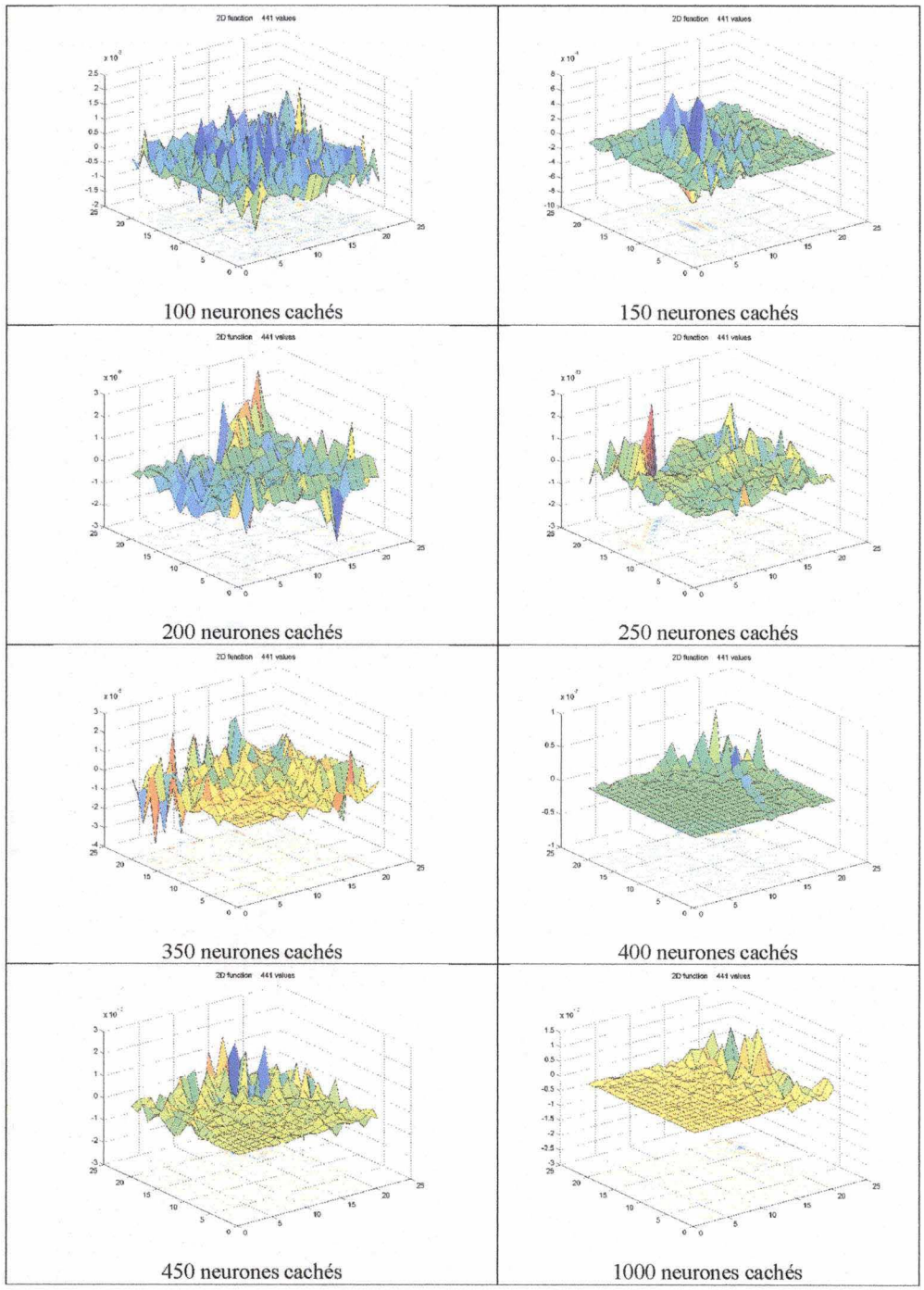


450 neurones cachés



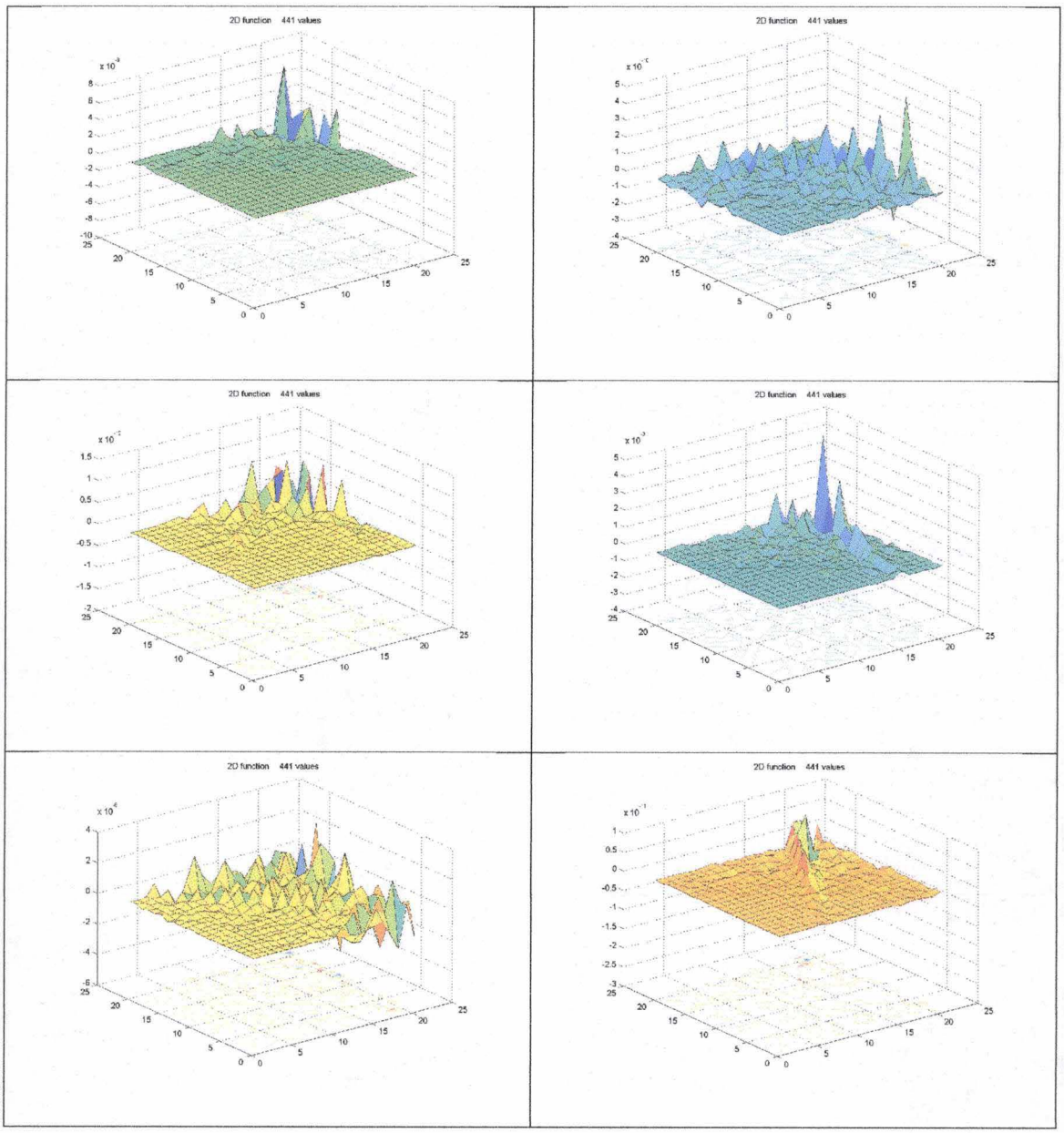
1000 neurones cachés

Etude de l'erreur globale résiduelle en fonction du nombre de neurones de la couche cachée

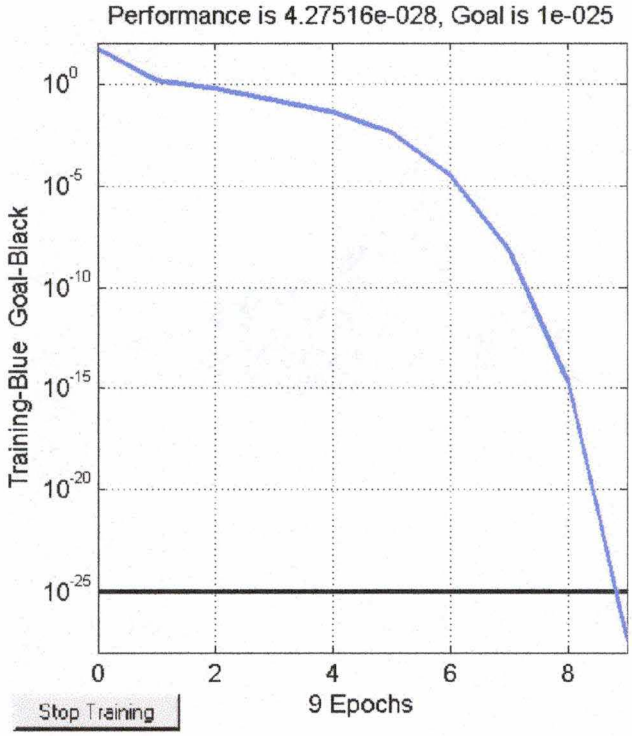


Les deux analyses montrent que vers 450 neurones, on atteint rapidement un niveau d'erreur résiduelle extrêmement faible : 10^{-28}, après une dizaine de périodes. Bien que le résultat soit déjà satisfaisant vers 250 neurones cachés, nous avons retenu comme paramètre de dimensionnement, un nombre de neurones égal à la taille de la table, soit 441. Ceci nous donne une règle simple pour travailler avec d'autres tailles de

tables. Nous avons ensuite examiné plus en détail, l'erreur résiduelle globale et l'erreur relative sur chaque point, pour un réseau à 441 neurones cachés. Les graphiques ci-après donnent la surface d'erreur, lors de six tests consécutifs, pour ce dimensionnement de réseau.



On constate que la convergence est rapide et que la précision obtenue est élevée. Il faut aussi remarquer que dans le cas présent, nous sommes dans une configuration de surapprentissage. Le graphique ci-après est répétitif. L'erreur globale est quasi nulle après neuf périodes, bien que les erreurs individuelles soient variables d'une exécution à la suivante.

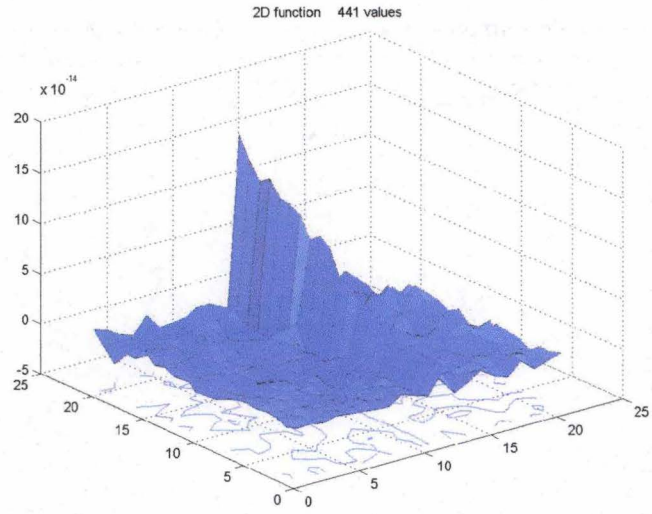


Analyse des erreurs ponctuelles relatives et absolues

Les valeurs reprises dans la table d'entrée varient d'un facteur de 1 à 200.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
1	0,67%	0,21%	0,05%	0,06%	0,10%	0,32%	0,01%	0,16%	0,19%	0,38%	0,40%	0,11%	0,21%	0,37%	0,37%	0,39%	0,35%	0,21%	0,32%	0,36%	0,21%
2	0,21%	0,56%	0,04%	0,04%	0,25%	0,14%	0,41%	0,38%	0,14%	0,07%	0,09%	0,04%	0,33%	0,18%	0,28%	0,30%	0,15%	0,26%	0,03%	0,31%	0,30%
3	0,07%	0,17%	0,60%	0,22%	0,34%	0,07%	0,43%	0,21%	0,37%	0,30%	0,28%	0,09%	0,43%	0,41%	0,13%	0,12%	0,16%	0,38%	0,06%	0,32%	0,14%
4	0,22%	0,24%	0,18%	0,44%	0,05%	0,16%	0,06%	0,24%	0,21%	0,15%	0,32%	0,30%	0,40%	0,13%	0,16%	0,35%	0,12%	0,21%	0,02%	0,06%	0,25%
5	0,05%	0,13%	0,14%	0,38%	0,79%	0,08%	0,38%	0,37%	0,14%	0,26%	0,43%	0,32%	0,12%	0,32%	0,27%	0,20%	0,36%	0,28%	0,40%	0,39%	0,09%
6	0,37%	0,18%	0,31%	0,12%	0,38%	0,61%	0,20%	0,33%	0,07%	0,09%	0,02%	0,37%	0,22%	0,39%	0,22%	0,04%	0,34%	0,36%	0,13%	0,07%	0,34%
7	0,44%	0,03%	0,10%	0,22%	0,07%	0,30%	0,61%	0,11%	0,25%	0,02%	0,20%	0,12%	0,03%	0,08%	0,12%	0,15%	0,42%	0,03%	0,38%	0,25%	0,11%
8	0,30%	0,08%	0,30%	0,01%	0,05%	0,11%	0,29%	0,58%	0,21%	0,06%	0,19%	0,04%	0,33%	0,36%	0,20%	0,08%	0,31%	0,26%	0,24%	0,10%	0,31%
9	0,19%	0,04%	0,13%	0,12%	0,03%	0,26%	0,30%	0,36%	0,80%	0,20%	0,00%	0,08%	0,06%	0,10%	0,27%	0,32%	0,09%	0,10%	0,23%	0,27%	0,33%
10	0,24%	0,05%	0,17%	0,10%	0,14%	0,33%	0,12%	0,02%	0,28%	0,85%	0,14%	0,29%	0,04%	0,05%	0,14%	0,05%	0,13%	0,08%	0,30%	0,06%	0,07%
11	0,09%	0,16%	0,23%	0,24%	0,33%	0,15%	0,16%	0,17%	0,40%	0,08%	0,87%	0,16%	0,18%	0,11%	0,19%	0,33%	0,17%	0,01%	0,41%	0,37%	0,06%
12	0,41%	0,29%	0,39%	0,07%	0,35%	0,42%	0,42%	0,00%	0,24%	0,33%	0,31%	0,44%	0,08%	0,26%	0,05%	0,29%	0,20%	0,39%	0,14%	0,04%	0,36%
13	0,36%	0,09%	0,12%	0,28%	0,12%	0,19%	0,08%	0,22%	0,33%	0,05%	0,13%	0,05%	0,92%	0,33%	0,25%	0,30%	0,26%	0,05%	0,29%	0,32%	0,19%
14	0,29%	0,09%	0,33%	0,30%	0,09%	0,44%	0,24%	0,11%	0,14%	0,24%	0,02%	0,30%	0,20%	0,63%	0,34%	0,24%	0,17%	0,13%	0,38%	0,34%	0,04%
15	0,26%	0,26%	0,11%	0,27%	0,23%	0,25%	0,37%	0,35%	0,04%	0,00%	0,02%	0,19%	0,30%	0,04%	0,71%	0,03%	0,11%	0,42%	0,31%	0,24%	0,34%
16	0,02%	0,05%	0,40%	0,34%	0,10%	0,26%	0,23%	0,01%	0,06%	0,17%	0,20%	0,15%	0,22%	0,07%	0,04%	0,45%	0,40%	0,29%	0,18%	0,10%	0,09%
17	0,38%	0,26%	0,06%	0,38%	0,27%	0,22%	0,00%	0,35%	0,42%	0,28%	0,22%	0,25%	0,08%	0,31%	0,22%	0,29%	0,61%	0,43%	0,33%	0,34%	0,41%
18	0,22%	0,13%	0,14%	0,28%	0,21%	0,33%	0,20%	0,03%	0,15%	0,30%	0,05%	0,37%	0,39%	0,37%	0,10%	0,23%	0,03%	0,60%	0,27%	0,04%	0,04%
19	0,07%	0,06%	0,27%	0,37%	0,07%	0,05%	0,01%	0,13%	0,32%	0,34%	0,42%	0,17%	0,13%	0,31%	0,26%	0,18%	0,17%	0,05%	0,67%	0,11%	0,33%
20	0,21%	0,36%	0,05%	0,18%	0,03%	0,04%	0,32%	0,21%	0,34%	0,27%	0,34%	0,11%	0,15%	0,32%	0,40%	0,04%	0,26%	0,04%	0,15%	0,61%	0,35%
21	0,39%	0,27%	0,36%	0,06%	0,04%	0,15%	0,15%	0,27%	0,14%	0,11%	0,17%	0,03%	0,42%	0,05%	0,10%	0,15%	0,14%	0,29%	0,15%	0,01%	0,71%

Les valeurs du test d'erreur sont représentées graphiquement dans la figure suivante :



Les erreurs individuelles sont détaillées dans la table ci-dessous. Elles sont multipliées par un facteur 10^{10}.

L'erreur absolue la plus grande concerne la case 12x12.
Elle est de 0,44 % x 9,11 x 10^{-10} = 4,0084 % x 10^{-10} = 0,00000000040084.

L'erreur relative la plus importante est celle de la cellule 12x21.
En valeur absolue, elle est de 0,03 % x 16,26 x 10^{-10} = 0,4878 % x 10^{-10} = 0,00000000004878

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
1	0,12	0,19	-0,07	0,25	0,11	0,01	-0,15	-0,24	0,66	0,21	-0,21	1,76	-0,57	0,12	0,97	0,01	-0,44	0,84	-0,29	0,16	0,51
2	-0,22	0,50	-0,25	0,29	-0,09	-0,01	0,26	0,02	0,19	0,43	-0,37	1,79	-0,16	0,50	0,34	0,04	0,03	1,01	0,71	-0,22	0,21
3	0,27	0,03	0,06	0,09	-0,04	-0,04	0,09	-0,24	0,62	0,19	-0,96	2,69	0,25	0,23	0,47	0,63	0,79	-0,58	0,33	0,00	0,31
4	0,39	0,12	0,01	-0,22	-0,09	-0,25	-0,30	0,14	-0,06	0,69	-0,45	2,91	0,04	0,68	-0,14	-0,34	0,28	-1,01	-0,44	0,00	-0,48
5	0,20	0,49	0,37	-0,17	0,40	-0,20	-0,04	0,05	0,49	-0,13	-0,08	2,24	-0,37	-0,66	0,38	0,46	-0,26	-0,44	0,06	-0,10	0,02
6	0,36	-0,02	0,07	0,43	0,28	0,67	0,08	-0,78	0,00	-0,66	-0,54	3,77	0,38	0,47	-0,21	-0,90	-0,24	1,57	0,39	0,15	0,31
7	0,24	0,00	-0,03	0,16	0,46	-0,08	0,52	-0,73	-0,39	-0,02	-0,19	3,83	0,62	0,15	-0,19	-0,08	-0,34	0,42	0,95	0,57	0,00
8	-0,22	-0,14	-0,63	0,23	0,17	-0,11	0,34	-0,37	-0,22	-0,95	-0,10	4,77	0,27	-0,17	0,28	0,07	1,37	0,51	1,20	0,01	0,61
9	-0,69	-0,27	-1,09	-0,17	-0,41	-0,82	-0,11	-1,08	-0,37	0,00	0,12	5,28	0,35	-0,03	-0,19	0,35	-0,11	-0,20	-0,72	0,00	-0,46
10	-0,72	-0,43	-0,16	0,20	-0,92	-1,07	-0,25	0,38	0,69	-0,55	-0,57	6,30	0,06	0,52	-0,58	0,35	-0,40	-0,62	0,91	-0,90	-0,72
11	-0,81	-0,06	0,38	-0,01	0,12	-0,06	-0,27	-0,34	0,21	-0,94	-1,54	6,43	-0,82	-0,07	-0,56	0,38	1,08	-0,06	0,46	-0,32	0,35
12	0,45	-0,29	-0,20	0,05	0,03	-0,06	-0,06	-0,04	0,75	-0,62	-0,11	9,11	-0,18	-0,82	1,15	0,52	-0,36	-0,15	-1,05	-0,13	-0,21
13	-0,25	-1,09	-1,48	0,11	-0,12	-0,01	-0,04	0,22	0,33	0,30	-1,51	9,89	0,09	-1,42	0,14	0,25	0,01	0,01	-0,09	0,63	0,35
14	0,03	-0,14	0,58	1,98	1,34	1,07	-1,16	-0,43	0,36	0,88	1,06	9,15	0,11	-0,80	0,91	-0,65	-0,48	0,22	-0,87	1,68	0,60
15	-0,01	-0,15	-0,40	-0,62	0,42	-0,17	-0,20	-0,54	-0,24	-0,12	-1,83	11,11	0,09	0,27	1,54	0,10	0,55	0,80	1,48	0,31	0,66
16	1,04	0,08	-0,01	-0,28	0,44	0,78	-0,29	-0,43	-0,75	-0,61	-1,14	11,76	0,02	-0,10	0,30	0,00	-0,32	-0,62	-1,96	-1,11	0,52
17	-0,24	-0,23	-0,83	0,01	-1,06	0,10	0,54	-0,72	-0,04	-0,74	-1,87	11,87	-1,03	-1,05	-1,11	-0,38	-1,92	-1,87	0,04	-0,64	-1,00
18	-0,02	-0,01	-0,71	-0,78	-0,02	0,24	0,64	0,68	0,60	-0,31	-3,02	13,31	-0,16	0,52	-1,93	0,00	-0,08	-0,15	-1,91	-2,72	-0,43
19	-1,66	-1,06	-1,07	-1,13	0,43	-1,16	-0,34	-0,37	-0,72	0,07	-1,97	1,26	0,00	0,02	0,56	0,24	1,17	-1,28	-0,72	-1,34	-1,97
20	-0,37	-0,16	-0,23	-1,04	-0,79	-0,34	-0,67	1,02	1,09	0,04	-1,75	14,17	-1,66	-1,38	2,42	1,29	-1,00	-0,70	-1,21	-0,60	-1,35
21	0,82	0,44	-0,38	-0,75	0,82	-0,69	-0,59	-0,04	0,27	-1,28	-2,79	16,26	0,60	-0,36	-0,19	1,03	1,09	-0,74	-1,22	-0,02	-1,06

F. Annexe F - Autres recherches sur les matrices de scores

Alignement d'acides aminés isolés

Cette nouvelle matrice est assez proche de la matrice BLOSUM 62, bien que construite sur une base d'information fort différente. Son originalité est de calculer pour les gaps un score, que nous appellerons « significatif », car il est établi avec les occurrences réelles des gaps enregistrés dans les alignements de la base de données HOMSTRAD.

Cette approche est originale par rapport aux références que nous avons trouvées dans la littérature, et qui se basent sur des règles empiriques pour déterminer le score d'une insertion ou d'une délétion. Cette approche est basée sur les exemples de la base de données HOMSTRAD, alors que celle qui est utilisée dans les matrices BLOSUM, donne un poids identique à tous les gaps, quel que soit l'acide aminé avec lequel le gap est aligné (se trouve en correspondance dans l'autre protéine).

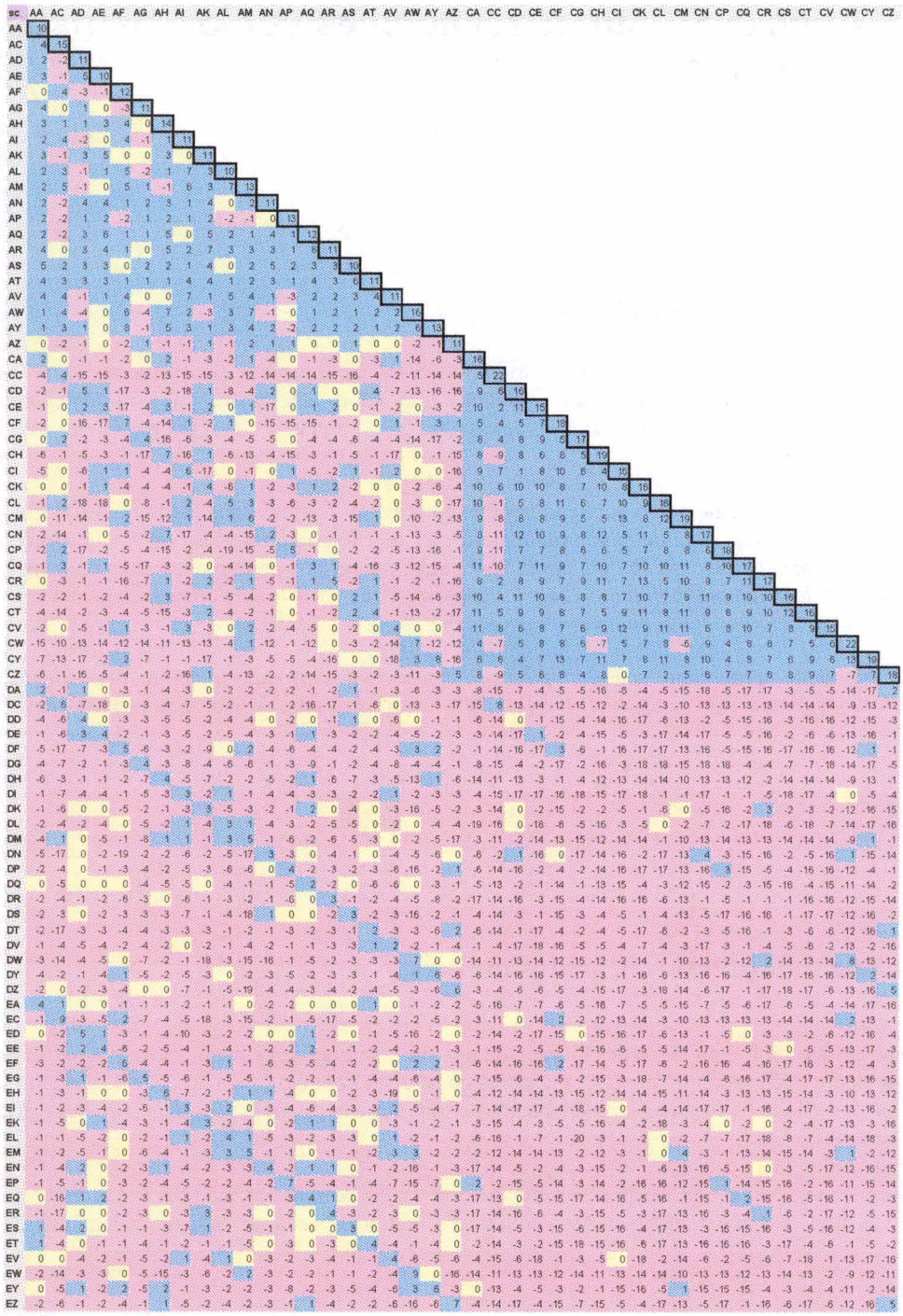
Les matrices de scores de la littérature sont construites en considérant chaque acide aminé, pris isolément par rapport à tous les autres. Ceci implique que l'on fait abstraction de l'influence des acides aminés voisins dans la chaîne.

Alignement de « doublets » d'acides aminés

Bien que ne relevant pas de l'application des réseaux de neurones, la question suivante s'est posée dans le cadre de cette étude : « l'utilisation de matrices de scores construites avec des blocs de deux acides aminés consécutifs, apporte-t-elle une précision supplémentaire par rapport aux matrices réalisées avec des acides aminés pris isolément.

Les programmes nécessaires la construction de cette matrice ont été développés. La matrice de dimension 441 (194 481 positions) a été transférée sur deux pages consécutives d'un tableur (maximum 256 colonnes).

Extrait de la matrice de scores de doublets



L'extrait ci-dessus est la première page de taille 42x84 positions de la matrice complète, dont la dimension est de 441x441 positions.

Les scores positifs sont en bleu, les scores négatifs en rouge et lorsque l'association de deux doublets est absente de la base de données, le score est mis à zéro pour indiquer que cette zone est non significative.

L'analyse de cette matrice de doublets nous laisse suggérer les conclusions suivantes :

- pour les doublets de deux acides aminés identiques (AA aligné avec AA), le score du doublet est proche de la somme des deux scores (A aligné avec A).
- pour les doublets possédant un même acide aminé en correspondance dans chaque doublet (GC aligné avec GA), le score est le plus souvent positif.
- pour les doublets inversés (CD<>KL et DC<>LK) les scores ne sont pas identiques.
- pour les doublets dont tous les acides aminés sont différents (CL<>AG), la somme des scores individuels (C<>A et L<>G) est fort différente du score du doublet.

Nous pouvons en déduire :

- que les zones de forte homologie vont avoir un poids significatif dans l'utilisation de cette matrice.
- que la matrice de score des doublets « devrait » être plus puissante que les matrices classiques, car chaque score contient plus d'information que la somme des scores individuels
- que son domaine d'utilisation est le même que celui de la matrice BLOSUM 62, ce qui était aussi le cas de la matrice que nous avons construite ci-avant sur base des acides aminés individuels.

Alignement de « triplets » d'acides aminés

La question de la précision apportée par les triplets n'a pas été développée. Bien que présentant un intérêt certain, elle nécessite de mettre en place des moyens informatiques puissants pour manipuler, exploiter et exporter des matrices de grande taille.

D'autre part, l'échantillon de la population sur lequel nous travaillons est relativement petit (1,7 millions d'informations) pour le nombre de classes à analyser (85,7 millions de classes).

Nous avons analysé la répartition des informations par classes pour les doublets, et appliqué à titre informatif ces proportions aux triplets, quadruplets,... . On constate que le nombre d'informations par classe est faible.

			taille population					
	nombre d' éléments	nombre de classes	1.675.000	> > 0	418.750	418.750	418.750	418.750
			100%	0,0%	25%	25,0%	25,0%	25,0%
Répartition de la population par classes			100%	32%	60%	6,5%	1,35%	0,15%
singleton	21	441	3.796	0	1.583	14.808	70.337	633.031
doublet	441	194.481	9	0	4	33	159	1.436
triplet	9.261	85.766.121	0,02	0	0,01	0,08	0,4	3,3
quadruplet	194.481	37.822.859.361	0,00004	0	0,00002	0,00017	0,0008	0,0074
quintuplet	4.084.101	16.679.880.978.201	0,0000001	0	0,00000004	0,00000039	0,000002	0,000017

Si on est proche de la limite de la faisabilité pour les triplets, les quadruplets et les quintuplets sont hors normes pour l'échantillon disponible (HOMSTRAD).

Gestion spécifique de fenêtres à trous dans MATCH-BOX

L'objectif est de faire une étude dans MATCH-BOX, d'alignements réalisés sur base de fenêtres à trous qui intègrent des « gaps » de style X_X_X ou X_X ou X__X.

Les tests réalisés précédemment avec les matrices BLOSUM n'avaient pas donné de résultats significatifs. Il pourrait être intéressant d'utiliser la matrice de score établie sur base d'alignements de doublets, qui intègre les gaps réels établis à partir de la base de données HOMSTRAD.

Par exemple, le score de A_A_A aligné avec GYGYG peut être calculé par la somme des scores : (A_<>GY) + (_A<>YG) + (A_<>GY) + (_A<>YG).

La réalisation de ce test devrait se faire en adaptant les modules de MATCH-BOX qui réalisent la manipulation des scores. Ces modules devraient également pouvoir identifier des doublets dans les séquences d'acides aminés des protéines.

10. Index des figures

11. Index des Tables

12. BIBLIOGRAPHIE

[1] Shigeo Abe and Takuya Inoue. *Fuzzy Support Vector Machines for Multiclass Problems* - in *European Symposium on Artificial Neural Networks*. 2002. Bruges (Belgium).

[2] D.A. Afonnikov, D.Y. Oshchepkov and N. A. Kolchanov, *Detection of conserved physico-chemical characteristics of proteins by analysing clusters of positions with co-ordinated substitutions.:* Bioinformatics, 2001, n° 17 (11): p. 1035-1046.

[3] S. Akaho, *VC Dimension theory for a learning system with forgetting:* Mathematical Informatics Sciences Laboratory, Umezono, Tsukuba-shi, Japan, 1985.

[4] Bruce Alberts, Dennis Bray and al, *Biologie Moléculaire de la Cellule*. Médecine-Sciences. 1996, Flammarion.

[5] Ethem Alpaydin, *Introduction to machine learning*. Adaptive computation and machine learning. 2004, Cambridge, Mass. : MIT Press.

[6] H. Altun and T. Yalcinoz. *Comparison of Genetic Algorithms, Hopfield and MLP Neural Network techniques for a constrained optimization problem* - in *International XII. Turkish Symposium on Artificial Intelligence and Neural Networks*. 2003. TAINN, Turkey.

[7] K. Ambos-Spies, S. Steven and U. Schöning, *Complexity Theory*. 1993, Cambridge University Press.

[8] Martin Anthony, *A Result of Vapnik with Applications:* Department of Statistical and Mathematical Sciences, London School of Economics, 1991.

[9] Martin Anthony and Norman Biggs, *PAC Learning and Artificial Neural Networks:* Department of Mathematics, London School of Economics and Political Science (University of London), 1994.

[10] Martin Anthony and John Shawe-Taylor, *A Suffcient Condition for Polynomial Distribution-Dependent Learnability:* Department of Mathematics, London School of Economics, London, 1995.

[11] A.N. Arslan and O. Egecioglu, *A new approach to sequence comparison : normalized sequence alignment.:* Bioinformatics, 2001, n° 17 (4): p. 327-337.

[12] A.N. Arslan and O. Egecioglu. *Approximation Algorithms for local alignment with length constraint* - in *The Latin American Theoretical Informatics Conference*. 2002. Cancun, Mexico.

[13] A.N. Arslan and O. Egecioglu, *Efficient Computation of Long Similar Subsequences.:* Department of Computer Science and Engineering - University of California., 2002.

[14] Christophe Assens, *Connexionnisme et théorie des organisations:* Actes du colloque sur la Recherche Neuronale en Sciences Economiques et de Gestion, 1995, n° 2: p. 193-206.

[15] Christian Attiogbé, *Eléments de complexité et de calculabilité:* Faculté des sciences et des techniques, Université de Nantes, 2001.

[16] T.K. Attwood, *The quest to deduce protein function from sequence: the role of pattern databases:* The International Journal of Biochemistry & Cell Biology, 2000, n° 32: p. 139-155.

[17] P. Baldi, L. Mazliak and P. Priouret, *Martingales et chaînes de Markov.* 2000, Hermann, Ed. des sciences et des arts.

[18] N. Barnier and al, *Combine & Conquer : Genetic Algorithm and Constraint Satisfaction Problem for optimisation:* Ecole Nationale de l'Aviation Civile, 1998.

[19] Nicolas Barnier and Pascal Brisset, *Optimisation par algorithme génétique sous contraintes:* Technique et science informatiques, 1999, n° 18 (1): p. 1-29.

[20] Vincent Barra, *Apprentissage:* Institut Supérieur d'Informatique, de Modélisation et de leurs Applications, Campus des Cézeaux, Aubière, France, 2006.

[21] Christian Barrett, *Investigation of non-pairwise proteins structure score functions using sets of decoy structure, PhD thesis*, in *Computer Sciences Department.* 2001, University of California.

[22] J. Barthélemy_Saint-Hilaire, *Logique d'Aristote, Tome II.* Edition Librairie de Ladrange (Paris). 1839, (Electronic Google Book).

[23] P. L. Bartlett, S. Boucheron and G. Lugosi, *Model Selection and Error Estimation:* Machine Learning,, 2002, n° 48: p. 85-113.

[24] A. D. Baxevanis and B.LF.F. Ouellette, *Bioinformatics.* 1998, Wiley-Interscience.

[25] Stephen D. Bay, *Nearest Neighbor Classification from Multiple Feature Subsets:* Department of Information and Computer Science, University of California, Irvine, California, 1999.

[26] Samy Bengio, *Statistical Machine Learning from Data:* IDIAP Research Institute, Martigny, Switzerland,, 2006.

[27] A. Berkaloff, J. Bourget and al, *Biologie et Physiologie Cellulaire.* 1967, Herman Paris.

[28] Cyrille Bertelle, Antoine Dutot, Frédéric Guinand and Damien Olivier, *Simulations distribuées par un algorithme fourmi:* Université du Havre, 2003.

[29] S. Bertolo, *Language Acquisition Learnability.* 2001, Cambridge University Press.

[30] Alberto Bertoni and B. Palano, *Structural Complexity and Neural Networks:* Dipartimento di Scienze dell'Informazione, Universita degli Studi di Milano, 2002.

[31] Jeff Bilmes, *What HMMs Can Do:* Department of Electrical Engineering, University of Washington, 2003.

[32] Anselm Blumer, Andrzej Ehrenfeucht, David Haussler and Manfred K. Warmuth, *Learnability and the Vapnik-Chervonenkis Dimension:* Journal of the Association for Computing Machinery., 1989, n° 36 (4): p. 929-965.

[33] Joel R. Bock and David A. Gough, *Predicting protein–protein interactions from primary structure:* Bioinformatics, 2001, n° 17 (5): p. 455-460.

[34] Rafal Bogacz and Christophe Giraud-Carrier, *A Novel Modular Neural Architecture for Rule-based and Similarity-based Reasoning:* Department of Computer Science, University of Bristol, UK, 1998.

[35] Alex Bollen, *L'ADN recombinant.* Institut de Biologie, Service de Génétique Appliquée. 1993, Université Libre de Bruxelles.

[36] B. Bouchon-Meunier, *La logique floue.* 1993, Presses universitaires de France.

[37] Hervé Bourlard and Samy Bengio, *Hidden Markov Models and other finite state automata for Sequence Processing:* Dalle Molle Institute for Perceptual Artificial Intelligence, IDIAP, Martigny, Switzerland, 2001.

[38] Olivier Bousquet, Stéphane Boucheron and G. Lugosi, *Introduction to Statistical Learning Theory:* Machine Learning, 2004, (LNAI 3176): p. 169-207.

[39] L.D. Brabandère, *Le Management des Idées.* 1998, Ed. Dunod.

[40] E.A. Breimer and M.K. Goldberg, *A Supervised Learning Approach for Detecting Significant Local Alignment.:* Rensselaer Polytechnic Institute - New York., 2002.

[41] P. Briffeuil, G. Baudoux, C. Lambert and al, *Comparative analysis of seven multiple protein sequence alignment servers: clues to enhance predictions reliability:* Bioinformatics, 1998, n° 14 (4): p. 357-366.

[42] Stephen H Bryant and Stephen F Altschul, *Statistics of sequence-structure threading:* Current Opinion in Structural Biology, 1995, n° 5: p. 236-244.

[43] David Bryant, Nicolas Galtier and M.-A. Poursat, *Likelihood calculation in molecular phylogenetics:* McGill Centre for Bioinformatics, Montreal, Canada, 2004.

[44] Z. Brzezniak and T. Zastawniak, *Basic Stochastic Processes.* 2003, Springer.

[45] Sergey Buldyrev, *Power Law Correlations in DNA Sequences:* Power Law, Scale-Free Networks and Genome Biology, 2006, n° (chapter 9).

[46] Christopher. J.C. Burges, *A Tutorial on Support Vector Machines for Pattern Recognition:* Data Mining and Knowledge Discovery,, 1998, n° 2: p. 121-167.

[47] Christopher Bystroff, Vesteinn Thorsson and David Baker, *HMMSTR: a Hidden Markov Model for Local Sequence-Structure Correlations in Proteins:* J. Mol. Biol., 2001, n° 301: p. 173-190.

[48] C.Z. Cai, W.L. Wang, L.Z. Sun and Y.Z. Chen, *Protein function classification via support vector machine approach:* Mathematical Biosciences, 2003: p. 111-122.

[49] Yu-Dong Cai, Shuo-liang Lin and Kuo-Chen Chouc, *Support vector machines for prediction of protein signal sequences and their cleavage sites:* Shanghai Research Centre of Biotechnology, Chinese Academy of Sciences, Shanghai , China, 2002.

[50] Yu-dong Cai and Shuo Liang Linb, *Support vector machines for predicting rRNA-, RNA-, and DNA-binding proteins from amino acid sequence:* Biochimica et Biophysica Acta, 2003, n° 1648: p. 127-133.

[51] Yu-Dong Cai, Xiao-Jun Liu and al, *Prediction of beta-turns with learning machines:* Peptides, 2003, n° 24: p. 665–669.

[52] C. A. Carson, J. M. Keller and al, *Escherichia coli O157:H7 Restriction Pattern Recognition by Artificial Neural Network:* Journal of Clinical Microbiology 1995, n° 33 (11): p. 2894-2898.

[53] Rich Caruana, Steve Lawrence and Lee Giles. *Overfitting in Neural Nets: Backpropagation, Conjugate Gradient, and Early Stopping* - in *Neural Information Processing Systems*. 2000.

[54] Gert Cauwenberghs and Tomaso Poggio, *Incremental and Decremental Support Vector - Machine Learning:* Massachusetts Institute of Technology, Cambridge, MA, 2002.

[55] Emilie Chanoni, Thierry Lecroq and Alexandre Pauchet. *Extraction de motifs dans des dialogues annotés par programmation dynamique*. in *Cinquièmes Journées Francophones Modèles Formels de l'Interaction*. 2009. Lannion, France.

[56] J.N. Chatain and A. Dussauchoy, *Systèmes experts*. 1987, Ed. Eyrolles.

[57] Michael Chau and Hsinchun Chen, *Comparison of Three Vertical Search Spiders.:* IEEE Computer Society, 2003, (0018-9162/03): p. 56-62.

[58] Jinmiao Chen and Narendra S. Chaudhari, *Protein Family Classification Using Second-Order Recurrent Neural Networks:* Genome Informatics, 2003, n° 14: p. 520-521.

[59] Vladimir Cherkassky, *Model complexity control and statistical learning theory:* Natural Computing,, 2002, n° 1: p. 109-133.

[60] Morten H. Christiansen, *Improving Learning and Generalization in Neural Networks through the Acquisition of Multiple Related Functions:* University of Southern California, Los Angeles, California, 1998.

[61] Y.J. Chung and al, *A MLP / HMM hybrid model using nonlinear predictors:* Speech Communication, 1996, n° 19: p. 307-316.

[62] Y.J. Chung and C.K. Un, *Multilayer perceptrons for state dependent weightings of HMM Likelihoods.:* Speech Communication, 1996, n° 18: p. 79-89.

[63] Piotr Ciskowski, *VC-Dimension of a Context-Dependent Perceptron.:* V. Akman et al. (Eds.): CONTEXT 2001,, 2001, n° (LNAI 2116): p. 429-432.

[64] David W. Coit and Alice E. Smith, *Using a Neural Network as a function for evaluator during GA search for reliability optiomization.:* Department of Industrial Engineering - University of Pittsburgh, 1996.

[65] Ronan Collobert and Samy Bengio, *A Gentle Hessian for efficient gradient descent:* IDIAP, Martigny, Switzerland, 2003.

[66] J.P. Comet, J.C. Aude and al, *Significance of Z-value statistics of Smith - Waterman scores for protein alignments:* Computers & Chemistry, 1999, n° 23: p. 317-331.

[67] P. Comon, *Classification supervisée par réseaux multicouches:* Traitement du Signal,, 1992, n° 8 (6): p. 387-407.

[68] Antoine Cornuéjols, Laurent Miclet and Y. Kodratoff, *Apprentissage artificiel*. 2002, Ed. Eyrolles.

[69] Michel Crucianu, Gilles Verley and al, *Un outil pour la comparaison et la validation d'algorithmes d'apprentissage à partir de données:* La Revue de Modulad,, 2000, n° 26: p. 55-69.

[70] Pierre Dagnelie, *Théories et Méthodes Statistiques*. 1973, Les Presses Agronomiques de Gembloux.

[71] Anthonin Danalet, *Les Premiers Analytiques d'Aristote*. 2007, Ecole Polytechnique Fédérale de Lausanne.

[72] R. David, K. Nour and C. Raffali, *Introduction à la logique*. 2001, Ed. Dunod.

[73] Renato De Mori, Michael Galler and Fabio Brugnara, *Search and learning strategies for improving Hidden Markov models:* Computer Speech and Language, 1995, n° 9: p. 107-121.

[74] J.P. Delahaye, *Outils logiques pour l'Intelligence artificielle*. 1986, Ed. Eyrolles.

[75] A.L. Delcher, S. Kasif and al, *Protein Secundary Structure Modelling with Probabilistic Networks.:* Dept. of Computer Sciences John Hopkins University, Baltimore., 1993.

[76] J.-L. Deneubourg, *Ant-Colony Optimization Algorithms (ACO):* Université Libre de Bruxelles (ULB), 1990.

[77] Jan Depenau, *Automated Design of Neural Network Architecture for Classification, Ph.D. Thesis - , in Faculty of Natural Sciences*. 1995, Aarhus University: Denmark.

[78] E. Depiereux and E. Feytmans, *Simultaneous and multivariate alignment of protein sequences: correspondance between physicochemical profiles and structurally conserved regions (SCR):* Prot. Engng., 1991, n° 4 (6): p. 603-613.

[79] E. Depiereux and E. Feytmans, *MATCH-BOX - A fundamentally new algorithm for the simultaneous alignment of several protein sequences:* Computer Applications in the Biosciences, 1992, n° 8 (5): p. 501-509.

[80] E. Depiereux, G. Baudoux, P. Briffeuil and al, *Match-Box_server: a multiple sequence alignment tool placing emphasis on reliability:* Computer Applications in the Biosciences, 1997, n° 13 (3): p. 249-256.

[81] Pedro Domingos, *The Role of Occam's Razor in Knowledge Discovery:* Data Mining and Knowledge Discovery, 1999, n° 3: p. 409-425.

[82] Randal Douc and Catherine Matias, *Propriétés asymptotiques de l'estimateur de maximum de vraisemblance pour des modèles de Markov cachés généraux:* C. R. Acad. Sci. Paris, 2000, n° 330 (1): p. 135-138.

[83] Olly Downs, *Learning Models for Continuous Nonnegative Data:* Hopfield Group, Princeton University, 2000.

[84] P.G. Drazin, *Nonlinear Systems*. 1992, Cambridge Texts in Applied Mathematics.

[85] G. Dreyfus, J-M. Martinez and al., *Réseaux de neurones*. 2002, Ed. Eyrolles.

[86] A.P. Dunmur and D.M. Titterington, *The influence of initial conditions on maximum likelihood estimation of the parameters of a binary hidden Markov model:* Statistics & Probability Letters, 1990, n° 40: p. 67-93.

[87] Elodie Duprat, *Caractérisation des domaines des superfamilles IgSF et MhcSF et classification fonctionnelle dans IMGT - Ph.D. Thesis* in *Bioinformatique*. 2005, Université de Montpelier II.

[88] N. Durand and al, *Neural Nets trained by Genetic Algorithms fort collision avoidance:* Ecole Polytechnique, Paris, 1997.

[89] B. Escofier and J. Pagès, *Analyses factorielles simples et multiples*. 1998, Ed. Dunod.

[90] Patricia Everaere, Sébastien Konieczny and Pierre Marquis. *Un point de vue épistémique sur la fusion de croyances : l'identification du monde réel.* in *Cinquièmes Journées Francophones Modèles Formels de l'Interaction*. 2009. Lannion, France.

[91] F. Famili, Z. Liu and al. *A Novel Data Mining Technique for Gene Identification in Time-Series Gene Expression Data.* in *16th European Conference on Artificial Intelligence*. 2004. Valencia, Spain.

[92] Piero Fariselli, Michele Finelli and al, *MaxSubSeq: an algorithm for segment-length optimization. The case study of the transmembrane spanning segments.:* Bioinformatics, 2003, n° 19 (4): p. 500-505.

[93] D. Foata and A. Fuchs, *Processus stochastiques*. 1998, Ed. Dunod.

[94] O. Francois and P. Leray, *Evaluation d'algorithmes d'apprentissage de structure pour les réseaux bayésiens:* INSA Rouen - Laboratoire PSI - FRE CNRS 2645, 2002.

[95] Olivier François and Philippe Leray, *Étude Comparative d'Algorithmes d'Apprentissage de Structure dans les Réseaux Bayésiens:* Laboratoire Perception, Systèmes, Information - insa-rouen, 2003.

[96] Olivier François and P. Leray, *Étude Comparative d'Algorithmes d'Apprentissage de Structure dans les Réseaux Bayésiens:* Laboratoire Perception, Systèmes, Information - FRE CNRS 2645, 2004.

[97] K. Fujarewicz, A. Swierniak and al, *Using Support Vector Machines for analysis of gene expression data from DNA microarrays:* Silesian University of Technology, 2004.

[98] J.-G. Ganascia, *Logique et Induction : un vieux débat.* Université Pierre et Marie Curie. 1999, Edition Cépaduès.

[99] E.J. Gardner, M. Simmons and D. Snustad, *Principles of Genetics*. 1991, John Wiley & Sons.

[100] Olivier Gascuel, S. Thiria, Y. Lechevallier and S. Canu, *Dimension de Vapnik-Chervonenkis et validité de l'approche neuronale* in "Statistique et Réseaux de Neurones". 1997, Edition Dunod.

[101] Daniel Gautheret, *Initiation à la Bioinformatique*. notes de cours. 2003, Marseille, Université de la Méditerranée.

[102] Dileep George and Jeff Hawkins, *A Hierarchical Bayesian Model of Invariant Pattern Recognition in the Visual Cortex:* Department of Electrical Engineering, Stanford University and Redwood Neuroscience Institute, Menlo Park, CA 94305, 2005.

[103] Dileep George and Jeff Hawkins, *Invariant Pattern Recognition using Bayesian Inference on Hierarchical Sequences:* Electrical Engineering, Stanford University and Redwood Neuroscience Institute, Menlo Park, California, 2005.

[104] Christophe Geourjon, *Bioinformatique structurale des protéines, PhD Thesis* - in *Pôle de BioInformatique Lyonnais,Laboratoire de Conformation des Protéines*. 2000, Université Claude Bernard - Lyon 1.

[105] Nick Goldman, Jeffrey L. Thorne and David T. Jones, *Using Evolutionary Trees in Protein Secondary Structure Prediction and Other Comparative Sequence Analyses:* J. Mol. Biol., 1996, n° 263: p. 196–208.

[106] Mirta B. Gordon, *Mémoire et apprentissage dans les systèmes artificiels:* ISSN : 1298-020X - © laboratoire Leibniz, 2003.

[107] Bernard Gosselin, *Application de réseaux de neurones artificiels à la reconnaissance automatique de caractères manuscrits - Ph.D. Thesis* in *Sciences Appliquées*. 1996, Faculté Polytechnique de Mons, Belgique.

[108] Y. Guermeur and C. Geourjon, *Apprentissage et bioinformatique:* LORIA & Pôle Rhône-Alpins de BioInformatique, Institut de Biologie et Chimie des Protéines, Lyon, 2003.

[109] Ping Guo, *Studies of Model Selection and Regularization for Generalization in Neural Networks with Applications - Ph.D. Thesis* in *Computer Science and Engineering*. 2001, The Chinese University of Hong Kong.

[110] Ping Guo, Michael R. Lyu and C. L. P. Chen, *Regularization Parameter Estimation for Feedforward Neural Networks:* Department of Computer Science, Beijing Normal University, China, 2002.

[111] B. Hammer and T. Villmann. *Mathematical Aspects of Neural Networks*. in *European Symposium on Artificial Neural Networks*. 2003. Bruges (Belgium).

[112] Aidan Harding, Mark Ryan and Pierre-Yves Schobbens. *Strategy synthesis without determinization*. in *Actes de la Conférence sur les Modèles Formels de l'Interaction*. 2003. Lille.

[113] Gilbert Hartman and Sanjeev Kulkarni, *Reliable reasoning, Induction and Statistical Learning Theory*. 2006, Jean Nicod Lectures, Centre National de la Recheche Scientifique.

[114] Jieyue He, Hae-Jin Hu and al, *Understanding Protein Structure Prediction Using SVM_DT:* Department of Computer Science, Southeast University, Nanjing, China, 2005.

[115] Steven Henikoff, *Scores for sequence searches and alignments:* Current Opinion in Structural Biology, 1996, n° 6: p. 353-360.

[116] S. Henikoff, J.G. Henikoff and P.C. Ng, *PHAT: a transmembrane-specific substitution matrix.:* Bioinformatics, 2000, n° 16 (9): p. 760-766.

[117] Thomas Henry, *Heuristiques pour l'apprentissage automatique de modèles biologiques par inférence grammaticale - PhD Thesis* in *Génomique et Informatique*. 2002, Université de Rennes-1.

[118] R. Herbrich, Thore Graepel and Robert C. Williamson, *The Structure of Version Space:* StudFuzz, 2006, n° 194: p. 257-274.

[119] William H. Hsu, Haipeng Guo and al, *A Permutation Genetic Algorithm for Variable Ordering in Learning Bayesian Networks from Data:* Laboratory for Knowledge Discovery in Databases, Kansas State University, 2002.

[120] Thomas Huber and Andrew E. Torda, *Protein Sequence Threading, the Alignment Problem, and a Two-Step Strategy:* Journal of Computational Chemistry, 1999, n° 20 (14): p. 1455-1467.

[121] Ioannis Iglezakis and Thomas Roth–Berghofer, *A survey regarding the central role of the case base for maintenance in case–based reasoning:* DaimlerChrysler AG, Research & Technology, 2000.

[122] Jean-Marie Jacquet and Koen De Bosschere. *On Relating Blackboards in the muLog Coordination Model.* in *Proceedings of The Thirtieth Annual Hawwaii International Conference on System Sciences.* 1997. Hawai.

[123] Anil K. Jain, Robert P.W. Duin and Jianchang Mao, *Statistical Pattern Recognition: A Review:* IEEE transactions Pattern Analysis and Machine Intelligence, 2000, n° 22 (1): p. 4-37.

[124] Martin Jambon, *Un système bioinformatique de recherche de similitudes fonctionnelles dans les structures 3D de protéines - Ph.D Thesis* in *Bioinformatique.* 2003, Université Claude Bernard - Lyon 1.

[125] Kyle Jensen, Mark Styczynski and Gregory Stephanopoulos, *Machine learning approaches to modeling the physiochemical properties of small peptides:* Department of Chemical Engineering, Massachusetts Institute of Technology, 2005.

[126] Jason M. Johnson and George M. Church, *Alignment and Structure Prediction of Divergent Protein Families: Periplasmic and Outer Membrane Proteins of Bacterial Efflux Pumps:* J. Mol. Biol., 1999, n° 287: p. 695-715.

[127] Marc Julia, *Mécanismes électroniques en Chimie Organique.* 1967, Gautier-Villars Paris.

[128] Y. Kamp and M. Hasler, *Réseaux de neurones récursifs pour mémoires associatives.* 1990, Presses polytechniques et universitaires romandes.

[129] Maricel Kann, Bin Qian and Richard A. Goldstein, *Optimization of a New Score Function for the Detection of Remote Homologs:* PROTEINS: Structure, Function, and Genetics, 2000, n° 41: p. 498-503.

[130] N. K. Kasabov, J. S. Kim, A. R. Gray and M. J. Watts, *FuNN - A Fuzzy Neural Network Architecture for Adaptive Learning and Knowledge Acquisition:* Foundation of Research Science and Technology (FRST) in New Zealand., 1996.

[131] Nikola Kasabov, *Evolving Fuzzy Neural Networks for Supervised/Unsupervised On-line, Knowledge-Based Learning:* IEEE Transactions of Systems, Man and Cybernetics,, 2001, n° 31 (6).

[132] Michael J. Kearns and U. V. Vazirani, *An Introduction to Computational Learning Theory.* 1994, MIT Press.

[133] Roni Khardon and Rocco A. Servedio, *Maximum Margin Algorithms with Boolean Kernels:* Department of Computer Science, Columbia University, New York., 2002.

[134] Bjarne Knudsen and Michael M. Miyamoto, *Sequence Alignments and Pair Hidden Markov Models Using Evolutionary History:* J. Mol. Biol., 2003, n° 333: p. 453-460.

[135] D.E. Knuth, *Fundamental Algorithms.* 1969, Addison Wesley.

[136] Pascal Koiran and E. D. Sontag, *Vapnik-Chervonenkis Dimension of Recurrent Neural Networks:* Laboratoire de l'Informatique du Parallélisme, Ecole Normale Supérieure de Lyon, 1997.

[137] Mark A. Kon and Leszek Plaskota, *Complexity of Predictive Neural Networks:* Boston University & Warsaw University, 1999.

[138] Ioannis Kopanas, Nikolaos M. Avouris and Sophia Daskalaki, *The Role of Domain Knowledge in a Large Scale Data Mining Project.* LNAI 2308, PP. 288-299, ed. I.P. Vlahavas and C.D. Spyropoulos. 2002.

[139] Anders Krogh, Björn Larsson and al, *Predicting Transmembrane Protein Topology with a Hidden Markov Model : Application to the Complete Genomes:* J. Mol. Biol., 2001, n° 305: p. 567-580.

[140] Sandor Kromesch and Sandor Juhász, *High Dimensional Data Visualization:* Department of Automation and Applied Informatics, Budapest University of Technology and Economics, 1998.

[141] S. Kwong, C.W. Chau and al, *Optimisation of HMM topology and its model parameters by genetic algorithms:* Pattern Recognition, 2001, n° 34: p. 509-522.

[142] Christophe G. Lambert, Nadia Léonard, Xavier De Bolle and Eric Depiereux, *ESyPred3D: Prediction of proteins 3D structures:* Bioinformatics, 2002, n° 18 (9): p. 1250-1256.

[143] Christophe Lambert, *Développement d'une méthode automatique fiable de modélisation de la structure tridimensionnelle des protéines par homologie et application au génome de Brucella melitensis - Ph.D. Thesis* in *Faculté des Sciences, Département de Biologie*. 2003, Facultés Universitaires Notre-Dame de la Paix: Namur.

[144] G. R. G. Lanckriet, M. Deng and al, *Kernel-Based Data Fusion and its application to Protein Function Prediction in Yeast:* Department of Biological Sciences & Department of Statistics, University of California, Berkeley, 2003.

[145] Georgios Lappas, *Estimating the Size of Neural Networks from the Number of Available Training Data:* J. Marques de Sa et al. (Eds.): ICANN 2007, 2007, n° Part I (LNCS 4668): p. 68–77.

[146] D. T. Larose, *Des données à la connaissance*. 2005, Ed. Vuibert informatique.

[147] R.H. Lathrop and T.F. Smith, *Global Optimum Protein Threading with Gap Alignment and Empirical Pair Score Functions:* M.I.T. Technology Licencing Office, 1996.

[148] Richard H. Lathrop. *On the Learnability of the Uncomputable* - in *Proceedings of the 13th International Conference on Machine Learning*. 1996. Bari, Italy.

[149] F. Lepage, *Eléments de la logique contemporaine*. 2001, Les presses de l'université de Montréal.

[150] Philippe Leray and Olivier François, *Réseaux Bayésiens pour la Classification - Méthodologie et Illustration dans le cadre du Diagnostic Médical:* INSA Rouen / PSI, FRE CNRS 2645, 2002.

[151] Na-na Li, Xiao-hui Niu, Feng Shi and Xue-yan Li, *Prediction of Protein Subcellular Locations Using Support Vector Machines:* School of Science, Huazhong Agriculture University, China, 2005.

[152] Jyh-Han Lin and Jeffrey Scott Vitter. *Complexity Results on Learning by Neural Nets* - in *Proceedings of the 2nd Annual ACM Workshop on Computational Learning Theory,*. 1989. Santa Cruz, CA.

[153] Kuang Lin, Jens Kleinjung, William R. Taylor and Jaap Heringa, *Testing homology with Contact Accepted mutatiOn (CAO): a contact-based Markov model of protein evolution:* Computational Biology and Chemistry, 2003, n° 27: p. 93-102.

[154] Qi Liu, Yi-Sheng Zhu and al, *A HMM-based method to predict the transmembrane regions of beta-barrel membrane proteins:* Computational Biology and Chemistry, 2002, n° 0: p. 1-8.

[155] Wynne Lock and C. Blouin, *Automated Editing of Multiple Protein Sequence Alignment Using Artificial Neural Networks:* Dalhousie University, Halifax, Canada, 2004.

[156] G. Luçotte, *Génétique moléculaire*. Collection Génie Génétique. 1985, Biofutur.

[157] R.B. Lyngso and al, *Metrics and similarity mesures for Hidden Markov Models:* AAAI, 1999.

[158] Wolfgang Maass, *Vapnik-Chervonenkis Dimension of Neural Net:* Institute of Theoretical Computer Science, Technische Universitaet Graz, Austria, 1995.

[159] David J. C. MacKay, *Bayesian Methods for Adaptive Models - Ph.D. Thesis* in *Mathematics*. 1992, Californian Institude of Technology, Pasadena, CA.

[160] David J.C. Mackay. *Density Networks and their Application to Protein Modelling*. in *Maximum entropy and Bayesian methods, Proceedings of the fourteenth international workshop*. 1994. Cambridge, UK.

[161] David J.C. Mackay, *Rate of Information Acquisition by a Species sujected to Natural Selection:* Cavendish Laboratory, Cambridge University, United Kingdom, 1999.

[162] Aleksander Malinowski, Tomasz J. Cholewo and Jacek M. Zurada. *Capabilities and Limitations of Feedforward Neural Networks with Multilevel Neurons* - in *Proceedings of the IEEE International Symposium on Circuits and Systems,*. 1995. Seattle, Wash., USA.

[163] R. ManKill and I. Koo, *Confidence Interval For Artificial Neural Networks with Incremental Learning:* Korea Advanced Institute for Science and Technology, 2002.

[164] Mario Marchand and Mostefa Golea, *On Learning Simple Neural Concepts: From Halfspace Intersections to Neural Decision Lists:* Ottawa-Carleton Institute for Physics, University of Ottawa, Canada, 1992.

[165] Mario Marchand and Mostefa Golea. *An Approximation Algorithm to Find the Largest Linearly Separable Subset of Training Examples*. in *Annual Meeting of the International Neural Network Society*. 1993. Ottawa.

[166] D. G. Martinson, W. Menke and J. R. Hopper, *An inverse approach to signal correlation:* Journal of Geophysics Research, 1982, n° 87: p. 4807-4818.

[167] H. Matsuda, T. Ishiham and A. Hashimoto, *Classifling molecular sequences using a linkage graph with their pairwise similarities:* Theoretical Computer Science, 1999, n° 210: p. 305-325.

[168] H. A. Mayer, *Symbiotic Coevolution of Arttificial Neural Networks and Training Data Sets.:* Department of Computer Science, University of Salzburg, Austria, 1998.

[169] D. A. McAllester, *PAC-Bayesian Stochastic Model Selection:* Machine Learning,, 2003, n° 51: p. 5-21.

[170] Kenneth McGarry, Stefan Wermter and John MacIntyre, *Hybrid Neural Systems: From Simple Coupling to Fully Integrated Neural Networks:* Neural Computing Surveys, 1999, n° 2: p. 62-93.

[171] P.L. Miller, *A critiquing Approach to Expert Computer*. 1984, Pitman Advanced Publishing Program.

[172] Tom. M. Mitchell, *Machine Learning*. 1997, McGraw-Hill.

[173] T. M. Mitchell, *Naïve Bayes and Logistic Regression:* Machine Learning,, 2005, (Chapter 1).

[174] Frantz Monfort, *Eléments de calcul des Probabilités et des Méthodes Statistiques*. Institut de Mathématiques. 1967, Université de Liège.

[175] Nicolas Monmarché, Christiane Guinot and Gilles Venturini, *Fouille visuelle et classification de données par nuage d'insectes volants:* École Polytechnique de l'Université de Tours - Département Informatique, 2002.

[176] Raymond J. Mooney, *A Preliminary PAC Analysis of Theory Revision:* Computational Learning Theory and Natural Learning Systems, 1995, n° 3: p. 43-53.

[177] Etsuko N. Moriyama and Junhyong Kim, *Protein Family classification with discriminant function analysis:* Department of Biology, University of Pennsylvania, 2002.

[178] K.R. Muller and al, *An Introduction to Kernel-Based Learning Algorithms:* IEEE Computer Society, 2001, n° 12 (2): p. 181-202.

[179] T. Muller and al, *Non-Symmetric Score matrices and the detection of Homologous Transmembrane Protein:* TBI, Heidelberg, Germany, 2001.

[180] Kevin Murphy and Saira Mian, *Modelling Gene Expression Data using Dynamic Bayesian Networks:* Life Sciences Division, Lawrence Berkeley National Laboratory Berkeley, California, 1999.

[181] Richard C. Murphy, *Phrase Detection and the Associative Memory Neural Network:* Computer Science and Engineering Department University of Notre Dame, San Diego, California., 1999.

[182] Marco Muselli and Francesca Ruffino, *Reliable Learning: A Theoretical Framework.* Traitement du Signal. Vol. III 2007, Eds. B. Apolloni et al. (Eds.). 174-183.

[183] Anatoli Nachev and Ivan Ganchev, *Learning with forgetting : an approach to achieve Adaptative Neural Networks:* MIS, Dept. A&F, National University of Ireland, Galway, Ireland, 2002.

[184] J-P. Nakache and J. Confais, *Approche pragmatique de la classification*. 2005, Ed. Technip.

[185] Kenta Nakai, Akinori Kidera and Minoru Kanehisa, *Cluster analysis of amino acid indices for prediction of protein structure and function:* Protein Engineering, 1988, n° 2 (2): p. 93-100.

[186] D. Naor and al, *Amino Acid Pair Interchanges at Spatially Conserved Locatiuons:* Molecular Biology Institute, University of California, Los Angeles, California, 1996.

[187] R. E. Neapolitan, *Learning Bayesian Networks*. 2004, Prentice Hall series in artificial intelligence.

[188] S. Needleman and C. Wunsch, *A general method applicable to the search for similarities in the amino acid sequence of two proteins:* J. Mol. Biol., 1970, n° 48: p. 443-453.

[189] A. Neumayer, *Molecular modeling of protein and mathematical prediction of protein structure:* Institut fur Informatik, Universitat Wien, Austria, 1997.

[190] Ziv Nevo and Ran El-Yaniv, *On Online Learning of Decision Lists:* Journal of Machine Learning Research, 2002, n° 3: p. 271-301.

[191] Jacques Nicolas, *SYstèmes et Modèles BIOlogiques, BIOinformatique et SEquences (projet Symbiose):* Rapport d'activité IRISA, 2002.

[192] G. Nicolis and I. Prigogine, *A la rencontre du complexe*. 1992, Presses Universitaires de France.

[193] Cédric Notredame, *Utilisation des Algorithmes génétiques pour l'analyse de séquences biologiques - Ph.D. Thesis* in *Bio−informatique*. 1998, Université Paul Sabatier: France.

[194] Fernando Santos Osório, *INSS : un système hybride Neuro-Symbolique pour l'Apprentissage Automatique - Ph.D. Thesis* in *Informatique*. 1992, Institut National Polytechnique de Grenoble - Laboratoire Leibniz-imag.

[195] Wassila Ouerdane. *Représentation des schémas d'arguments dans un contexte d'aide à la décision multicritère*. in *Cinquièmes Journées Francophones Modèles Formels de l'Interaction*. 2009. Lannion, France.

[196] Jong Park, Kevin Karplus and al, *Sequence Comparisons Using Multiple Sequences Detect Three Times as Many Remote Homologues as Pairwise Methods:* J. Mol. Biol., 1998, n° 284: p. 1201±1210.

[197] Jan Poland and Andreas Zell, *Different Criteria for Active Learning in Neural Networks: A Comparative Study:* University of Tübingen, Germany, 1997.

[198] Robert Pollock, Toby Lane and Michael Watts, *A Kohonen Self-Organizing Map for the functional classification of proteins based on one-dimensional sequence information:* Department of Biochemistry, University of Otago, Dunedin, New Zealand, 2002.

[199] I. Prigogine, *Les lois du chaos*. 1993, Ed. Flammarion.

[200] S. Primrose, R. Twyman and R. Old, *Principes de Génie Génétique*. 2004, De Boeck & Larcier.

[201] L. Pun, *Systèmes industriels d'intelligence artificielle*. 1984, Edi Tests (P.S.I.).

[202] Jianding Qiu, Ruping Liang, Xiaoyong Zou and Jinyuan Mo, *Prediction of protein secondary structure based on continuous wavelet transform:* Talanta, 2003, n° 61: p. 285-293.

[203] P. P. Raghu and B. Yegnanarayana, *Supervised Texture Classification Using a Probabilistic Neural Network and Constraint Satisfaction Model:* IEEE Transactions on Neural Networks., 1998, n° 9 (3): p. 516-522.

[204] Ramasamy Ramachandran and Natarajan Gunasekaran, *Optical Implementation of Two Dimensional Bipolar Hopfield Model Neural Network.:* Proc. Natl. Sci, Counc. ROC(A), 2000, n° 24 (1): p. 73-78.

[205] Vincent Ranwez, *Méthodes efficaces pour reconstruire de grandes phylogénies suivant le principe du maximum de vraisemblance - Ph.D. Thesis* in *Informatique*. 2002, Université de Montpellier II.

[206] J. David Rawn, *Traité de Biochimie*. 1990, De Boeck.

[207] François Rechenmann, *Projet helix - Informatique et génomique.:* Institut National de recherche en Informatique et en Automatique (INRIA), 2002.

[208] Jean-Philippe Rennard, *Genetic Algorithm Viewer : Démonstration d'un algorithme génétique:* Introduction aux Algorithmes génétiques - (file:///M|/gavpdfw.html). 2000.

[209] Isidore Rigoutsos, Aris Floratos and al, *The Emergence of Pattern Discovery Techniques in Computational Biology:* Metabolic Engineering, 2000, n° 2: p. 159-177.

[210] Enrique Romero and R. Alquézar, *Maximizing the Margin with Feed-forward Neural Networks:* IEEE Computer Society -, 2002, (0-7803-7278-6/02).

[211] Mark Ryan and Pierre-Yves Schobbens. *Intertranslating counterfactuals and updates*. in *12th European Conference on Artificial Intelligence*. 1996. Budapest, Hungary.

[212] Robert E. Schapire. *A Brief Introduction to Boosting* - in *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence,*. 1999. AT&T Labs, Shannon Laboratory, NJ, USA.

[213] Tobias Scheffer, *Error Estimation and Model Selection - Doktor der Naturwissenschaften* in *Informatik*. 1999, Technischen Universität Berlin.

[214] Michael Schmitt, *An Improved VC Dimension Bound for Sparse Polynomials:* Lehrstuhl Mathematik und Informatik, Fakultät f¨ur Mathematik Ruhr-Universität Bochum, Germany, 2004.

[215] Agnes Schumann, *Neural Networks versus Statistics : a comparative study of their classification performance on well log data:* Geoinformatics, Free University of Berlin, Germany, 1994.

[216] M. Seeger, *PAC-Bayesian Generalisation Error Bounds for Gaussian Process Classification:* Journal of Machine Learning Research, 2002, n° 2: p. 233-269.

[217] Matthias Seeger, *Bayesian Gaussian Process Models: PAC-Bayesian Generalisation Error Bounds and Sparse Approximations - Ph.D. Thesis* in *Informatics*. 2003, Institute for Adaptive and Neural Computation, University of Edinburgh.

[218] Enrique Carlos Segura, *On the relation between Entropy, Information and Stability in Hopfield memories.:* Departamento de Computación - Facultad de Ciencias Exactas y Naturales - Universidad de Buenos Aires, 1986.

[219] C. E. Shannon, *A Mathematical Theory of Communication:* The Bell System Technical Journal,, 1948, (27): p. 379–423, 623–656,.

[220] John Shawe-Taylor, Martin Anthony and N.L. Biggs, *Bounding Sample Size with the Vapnik-Chervonenkis Dimension:* Department of Computer Science - Royal Holloway and Bedford New College, 1989.

[221] Y. Shirai and J.-I. Tsujii, *Intelligence artificielle*. 1987, Ed. Eyrolles.

[222] TF. Smith and MS. Waterman, *Identification of common molecular subsequences:* J Mol Biol., 1981, n° 147 (1): p. 195-197.

[223] L. Smith, L. Yeganova and W. J. Wilbur, *Hidden Markov models and optimized sequence alignments:* Computational Biology and Chemistry, 2003, n° 27: p. 77-84.

[224] Société du Nouveau Littré, *Le Petit Robert, Dictionnaire alphabétique et analogique de la langue française:* 1972.

[225] S. Sonnenburg, G. Rätsch, A. Jagota and K.-R. Müller, *New Methods for Splice Site Recognition:* University of Berlin & Canberra & Santa Cruz, 2003.

[226] William M. Spears and Kenneth A. De Jong, *Using Neural Networks and Genetic Algorithms as Heuristics for NP-Complete Problems:* Naval Research Laboratory, Washington & George Mason University, Fairfax., 1990.

[227] Christian Spronk, Jens Linge, Cornelis Hilbers and Geerten Vuister, *Improving the quality of protein structures derived by NMR spectroscopy:* Journal of Biomolecular NMR, 2002, n° 22: p. 281-289.

[228] David T. Suzuki, *L'Analyse Génétique*, ed. U. o. B. Columbia. 1991, De Boeck.

[229] S. Thiria, Y. Lechevazlier and et al., *Statistiques et méthodes neuronales*. 1997, Ed. Dunod.

[230] Julie D. Thompson, Frédéric Plewniak and al, *Towards a Reliable Objective Function for Multiple Sequence Alignments.:* J. Mol. Biol., 2001, n° 314: p. 937-951.

[231] J. Tisdall, *Introduction à Perl pour la bioinformatique*. 2001, Ed. O'Reilly.

[232] Michael L. Tress and David Jones, *Predicting Reliable Regions in Protein Alignments from Sequence Profiles.:* J. Mol. Biol., 2003, n° 330: p. 705-718.

[233] E. P. K. Tsang and C. J. Wang, *A Generic Neural Network Approach For Constraint Satisfaction Problems*. Neural Network Applications, ed. E. J.G.Taylor. 1992. p. 12-22.

[234] R. Turner, *Logiques pour l'intelligence artificielle*. 1984, Ed. Masson.

[235] Gabor E. Tusnady and Istvan Simon, *Principles Governing Amino Acid Composition of Integral Membrane Proteins: Application to Topology Prediction:* J. Mol. Biol., 1998, n° 283: p. 489-506.

[236] Dominique Urbani, *Méthodes statistiques de sélection d'architectures neuronales : application à la conception de modèles de processus dynamiques, Ph.D. Thesis* in *Microélectronique et microinformatique*. 1995, Université Paris VI.

[237] L.G. Valiant, *A theory of the learnable:* Communications of the ACM, 1984, n° 27 (11): p. 1134-1142.

[238] Hans Vandierendonck, Jean-Marie Jacquet, Bavo Nootaert and Koen De Bosschere, *A Formal Model for Microprocessor Caches:* Institut d'Informatique FUNDP, 2008.

[239] V. N. Vapnik, *The Nature of Statistical Learning Theory*. 1995, Springer.

[240] V. N. Vapnik, *Statistical Learning Theory*. 1998, Wiley.

[241] Nicolas Vayatis, *Inégalités de Vapnik-Chervonenkis et mesures de complexité - Ph.D. Thesis*, in *Mathématiques appliquées*. 2000, Ecole Polytechnique, PARIS.

[242] Alain Vergnoux, *L'explication dans les sciences*. 2003, Edition De Boeck.

[243] Richard H. Wade, *Sequence Landmark Patterns Identify and Characterize Protein Families:* Structure, 2002, n° 10: p. 1329-1336.

[244] Jason T. L. Wang, Qicheng Ma, Dennis Shasha and Cathy H. Wu, *Application of Neural Networks to Biological Data Mining: A Case Study in Protein Sequence Classification:* National Biomed. Research, Foundation (NBRF-PIR), Georgetown Univ. Medical Ctr., Washington, 2001.

[245] Manfred K. Warmuth. *Towards Representation Independence in PAC Learning* - in *Proceedings of the International Workshop on Analogical and Inductive Inference*. 1989. Rheinhardsbrunn Castle, German Democratic Republic.

[246] Geoffrey I Webb, *Generality is more significant than complexity: Toward an alternative to Occam's Razor:* School of Computing and Mathematics, Deakin University, Geelong, Australia, 1994.

[247] Olaf Weiss and Hanspeter Herzel, *Mesuring Correlations in Protein sequences:* Institute for Theoretical Physics, Humbold University Berlin, Germany, 1995.

[248] Olaf Weiss and Hanspeter Herzel, *Correlations in protein Sequences and Property Codes:* Institute for Theoretical Biology, Humbold University Berlin, Germany, 1997.

[249] D.R. Westhead, J.H. Parish and R.M. Twyman *Bioinformatics*. 2002, BIOS Scientific Publishers Limited

[250] Cathy H. Wu, Hsi-Lien Chen, Chin-Ju Lo and Jerry W. McLarty, *Motif Identification Neural design for rapid and sensitive Protein Family search:* The University of Texas Health Center, Tyler, TX, 1996.

[251] An-Suei Yang and Barry Honig, *An Integrated Approach to the Analysis and Modeling of Protein Sequences and Structures. I. Protein Structural Alignment and a Quantitative Measure for Protein Structural Distance:* J. Mol. Biol., 2003, n° 301: p. 665-678.

[252] An-Suei Yang and Barry Honig, *An Integrated Approach to the Analysis and Modeling of Protein Sequences and Structures. II. On the Relationship between Sequence and Structural Similarity for Proteins that are Not Obviously Related in Sequence:* J. Mol. Biol., 2003, n° 301: p. 679-689.

[253] G. Yona and M. Levitt, *Towards a complete map of the protein space based on a unified sequence and structure analysis of all known proteins:* American Association for Artificial Intelligence, 2000.

[254] Gabriele Zenobi and Pádraig Cunningham, *An Approach to Aggregating Ensembles of Lazy Learners that Supports Explanation:* Department of Computer Science, Trinity College Dublin, 2001.

[255] Jie Zheng, Lalitha Vankataramanan and Fred J. Sigworth, *Hidden Markov Model Analysis of Intermediate Gating Steps Associated with the Pore Gate of Shaker Potassium Channels:* J. Gen. Physiol. The Rockefeller University Press, 2001, n° 118: p. 547-562.

[256] Jean-Daniel Zucker and Yann Chevaleyre, *Comprendre et résoudre les problèmes d'apprentissage multi-instances et multi-parties:* LIP6-CNRS, Université Paris VI, 1998.